PROGRAMMED INSTRUCTION TEXT PROCESSING USING GRAMMATICAL ANALYSIS

A Thesis Submitted
In Partial Fulfilment of the Requirements
For the Degree of
MASTER OF TECHNOLOGY

by

R. GANGADHARAN

to the

Department of Electrical Engineering
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

August 1972

CERTIFICATE

This is to certify that the thesis entitled, "Programmed Instruction Text Processing Using Grammatical Analysis" is a record of the work carried out under my supervision and that it has not been submitted elsewhere for a degree.

H.V. Sahasrabuddhe

Dr. H.V. Sahasrabuddhe
Assistant Professor
Department of Electrical Engineering
Indian Institute of Technology, Kanpur

Kanpur
August 1972

ACKNOWLEDGEMENT

I am deeply indebted to my thesis advisor, Dr. H.V. Sahasrabuddhe, for his invaluable guidance and encouragement throughout this work. I am grateful to my colleague, Mr. S. Venkataraman for many fruitful discussions and sustained help in accomplishing this thesis work. I thank the staff of the Computer Centre, Indian Institute of Technology, Kanpur for the facilities provided on IBM 7044. Thanks with particular appreciation are also acknowledged to my colleagues, Mr. Kannan, for his assistance in running the programs and Mr. P. Chinnuswamy for his help in debugging the programs during initial stages. Mr. K.N. Tewari deserves special mention for the excellent typing job.

R. Gangadharan

TABLE OF CONTENTS

needs to be extracted, translated, sorted and so on. That means, in some way, the computer understands what the natural language data means, in the same way as man does. Man arrives at the content of the natural language text by checking against what he already knows about the text under consideration, whereas, a computer has to resort to previously stored information (a dictionary of basic words) for processing the natural language text data. A natural language data processor provides an organised dictionary of basic words and an elaborate grammatical analysis using codings, functional units etc., which aid the computer in processing the contents of the input text for various applications. One such application mentioned below, is the main interest of this report. The high speed, the high reliability, the capability of performing logical operation, the capability of handling huge bulks of data, the adaptability to a wide vatiation of data etc. make an automatic computer, useful for various applications in the field of natural language data processing.

1.2 AN APPLICATION OF NATURAL LANGUAGE DATA PROCESSING

The field of application of natural language data processing is a vast one[2,3]. P.L.Garvin[2] points out that natural language data processing serves two purposes, (i) a linguistic analysis, to obtain analytic linguistic results, (ii) information handling, wherein the validity of the analytic linguistic results can be tested. Language

data processing for information handling includes two fields, (a) machine translation from one language to another, (b) information storage and retrieval, automatic abstracting, indexing and classification of documents, question answering, analysis of writing styles, all of which can be grouped as content processing. This report deals with a slightly different application in the field of content processing viz. guessing responses to programmed instruction texts. A preliminary processing of the programmed text data provides a grammatical coding to the words in the text. A further analysis on the coded text, provides the clue for obtaining the desired response.

The programmed instruction text considered, for our purposes, is a text of few sentences, each conveying a meaning. At the end of the text, a sentence with a blank entry occurs. We call this the "question sentence". After obtaining the grammatical coding for the "question sentence", a frame of words is selected from it according to their grammatical coding. We call this the "test frame". The "test frame" is used to pick from the text, appropriate sentences which are likely to contain the desired response to the question sentence. Depending on the grammatical coding of the blank entry in the "question sentence", a suitable analysis is carried out to pick the correct response from the selected sentences.

Chapter II describes the approach used with the help of a overall block diagram.

Chapter III describes the various routines with the help of flow charts.

Chapter IV concludes the report with few remarks on the results obtained.

CHAPTER II

AN APPROACH TO PROGRAMMED TEXT PROCESSING

2.1 DESCRIPTION OF THE SCHEME

The main aim has been to develop a scheme by which a computer guesses the desired response to programmed texts using only the grammatical analysis. The approach to programmed text processing, is explained with the help of an overall block diagram given in Figure 2.1.

2.1.1 Input Phase

As an input to the programmed text processor, a programmed instruction manual for teaching programming techniques to beginners is considered. The text has to be read and scanned characterwise to separate it into sentences. Each sentence, then, has to be stored wordwise for analysis. This constitutes the first stage of the scheme, indicated by the first block in Figure 2.1.

2.1.2 Preliminary Grammatical Analysis

Just as a beginner must know some basic words for understanding the manual under discussion, so also, a computer requires the development of an organised dictionary of words. And such a dictionary is built by conducting "a frequency of occurrence of words" analysis on texts. The response enabled us to arrive at a 1000 word dictionary which best suits our text processing. By introducing

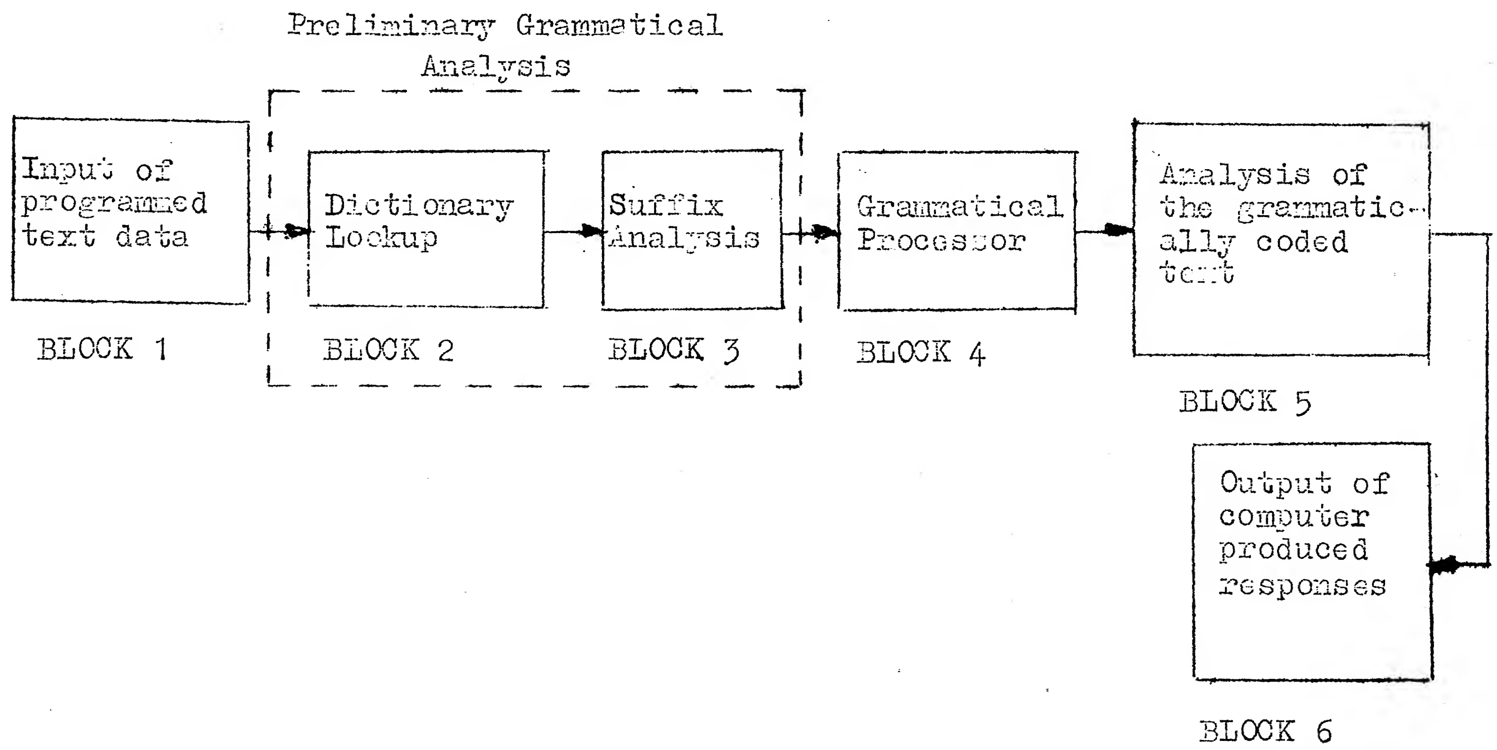


Figure 2.1: OVERALL BLOCK DIAGRAM FOR THE SCHEME

grammatical informations to each entry in the dictionary, additional restriction is placed on the simple word by word match procedure, which is used to obtain the desired computer response to programmed texts. The grammatical codes assigned to a dictionary entry indicate its position in different sentence structures. The positions occupied by the attributes, prepositions, verb qualifiers, adjective and adverb qualifiers etc. are classified as "functional units of the sentence". For example, some functional units are given below.

Functional Unit	Example
A (attribute)	a, an, the, that etc.
B (verb qualifiers)	would, be,may etc.
F (prepositions)	from, on, in etc.

The other word positions in a sentence are classified as "form class units". These positions include nouns, verbs, adverbs and adjectives. For example,

Form class Unit	Example
1 (nouns & pronouns)	book, John, that etc.
2 (verbs)	is, was, be, were etc.
3 (adjectives)	good, bad etc.
4 (adverbs)	away, around etc.

The coding of each entry in the dictionary is filled with the help of a conventional dictionary. Since the same word is likely to take different positions in different sentences, multiple coding (a combination of the above coding units) is

provided for some dictionary entries. For example, the word 'THAT' can appear either as a pronoun or as an attribute and hence, given the code 'A1' in the dictionary.

A look at the nature of the text revealed that some words do not occur as basic words, but as compound words with prefixes and suffixes. The natural tendency is to include such prefixed and suffixed words also in the dictionary and this will make the dictionary not a limited class one, as we intended it to be. By limited class dictionary, we mean the one with basic words only. The approach adopted to take care of the compound words in texts is to perform preliminary prefix and suffix tests, on the words. These tests remove the prefix and suffix of a word, identify them with standard prefixes and suffixes, and then attempt to search for the root of the word in the dictionary.

As the occurence of prefixed words is not very common in texts and also as the prefixes do not carry any meaning in the grammatical sense, only a suffix analysis of words is considered[6]. To distinguish a compound word from a basic word, one has to determine the actual suffix present in the compound word. This necessitates, a systematic suffix analysis, using commonly occuring suffixes or standard suffixes. The suffixes having the same length (in terms of characters) are grouped together to simplify the analysis. Each group is tried in order to determine the actual suffix of the given word. We know that a compound word or a derived word,

derives its grammar category from its suffix. For example, a basic word which might be a noun, will become a verb or an adverb or an adjective depending on the suffix with which it is used. So by assigning a grammatical code (s) to each of the standard suffixes, the grammatical coding of the compound word can be determined, provided its suffix has already been identified with one of the standard suffixes.

However, a suffix may not provide a unique grammar assignment to the compound word. A dictionary look up of the basic word from which the compound word is derived, resolves this difficulty. This requires, that the basic word or root of the compound word must be determined first. Depending upon the suffix, a basic word gets altered to form the derived word. So a mere separation of the suffix from the derived word, may not give its basic word. One needs to do addition or deletion of characters to or from the root obtained after the suffix has been removed from the derived word. Since, different suffixes demand different operations to be done on the root, each suffix must have in addition to the grammar code, an action code also to specify the operation.

The action codes are numbered from 1 to 8. The codes and the action specified by each on the root of the given word are given below.

Code Number	Action
1	no action
2	Add 'E' to the root

Code Number	Action
3	Add 'Y' to the root
4	Delete last character from the root
5	Delete last character and add 'E' to the root
6	Delete last character and add 'Y' to the root
7	Add 'ITY' to the root
8	Add 'D' to the root

Since some suffixes may demand more than one operation, they are provided with a combination of the above action codes. Still, for some suffixes, which do not fall in the general category described above, it is necessary to do a special analysis.

Once the basic word is obtained using the action codes of the suffix identified, one has to determine the grammatical coding of the basic word by a dictionary lookup. This dictionary lookup provides a grammatical coding only if the basic word matches an entry in the dictionary. To obtain a unique code assignment for the compound word under discussion, the intersection of the two codings, one provided by its suffix (a multiple code assignment), and the other provided by its root from a dictionary lookup, is selected. This intersection, in most cases provides a unique code assignment. The above analysis, termed as the

preliminary grammatical analysis, is indicated by the blocks 2 and 3 in Figure 2.1. The dotted line from block 3 to block 2 refers to the dictionary search required by the suffix test.

2.1.3 Detailed Grammatical Analysis

A preliminary grammatical analysis may not be able to assign any grammatical coding to some words or it may assign, still, multiple coding to some words. This difficulty is overcome, by providing a grammatical processor (block 4 in Figure 2.1). With the help of the functional units, assigned to some words in the input sentence, it guesses the possible grammar code of the words, for which no assignment has been made previously. By making a second scan of the sentence and by making use of the uniquely assigned grammar codes of some words in the sentence, the grammatical processor tries to eliminate multiple coding assignments. Since the grammatical processor[5] works only on a heuristic notion, multiple coding assigned to some words may still remain unresolved.

2.1.4 Programmed Text Processing

The grammatical coded text, now, needs to be analysed to guess the desired response to the sentence with a blank entry (termed as "question sentence"), which occurs at the end of the programmed text. The grammatical coding of the question sentence helps to pick out a frame of words, having specific grammatical units assigned to them (termed as "test frame"). The test frame is then matched word by word

with the words of the sentences in the text. This matching picks out a sentence, which is most likely to give the desired response. Using the grammatical coding of the selected sentence, its words are grouped, if they fall within a specific grammar frame. These frames are called "group frames of the sentence". A similar group frame for the blank entry is also determined. Depending on the grammatical unit, guessed at the blank entry, these group frames provide separate analysis to choose the appropriate word in the selected sentence, (words not present in the test frame) as the desired response. This programmed text analysis is shown by block 5 in Figure 2.1.

2.1.5 Output Phase

The final stage of the scheme (block 6 in Figure 2.1) is printing the computer response to the question sentence in the programmed text.

The complete block diagram has given us an overview of the flow of the input text through the various stages of the scheme described above and forms an introduction to the detailed description of the program given in the next chapter.

2.2 LANGUAGE USED AND CORE ORGANISATION

The complete program, which implements the scheme on IBM 7044 in IIT/Kanpur, is written in FORTRAN IV. Some aspects of the program, such as, bit manipulation for shifting the contents of a register or for packing

information from other registers into a word etc.,could have been easily achieved, if the assembly language of IBM 7044, MAP, had been used for some subroutines. However, in order to have a machine independent program developed, the main program and all its subroutines have been written in FORTRAN IV. Some system features of IBM 7044 like FILE 99, used for format conversion, could always be dispensed with, by writing one's own routines. No usage of backups like discs, tapes, etc., is made as they tend to increase the overall execution time.

IBM 7044 computer has a total core memory of 77K (octal). System, including IOCS, occupies a core of 12.3K (octal). Input and output buffers take 1.5K (octal). The core occupied by the object program, which includes a main routine, 10 subroutines, and system supplied routines is 52.3K (octal). The program's dictionary and suffixes, given as data, are provided a dimension of 9.1K (decimal). Only 4.1K (decimal)are in present use, the rest being provided for extending the dictionary and suffix with additional information. The grammar codes of a dictionary entry are packed in a single word, one for each byte of the word. The 6 bytes in a word provide for six codes for each dictionary entry. A total of 64 different codes are possible, of which, only 14 different codes are in use. Codes are stored in bytes to simplify format conversion and other manipulations.

CHAPTER III

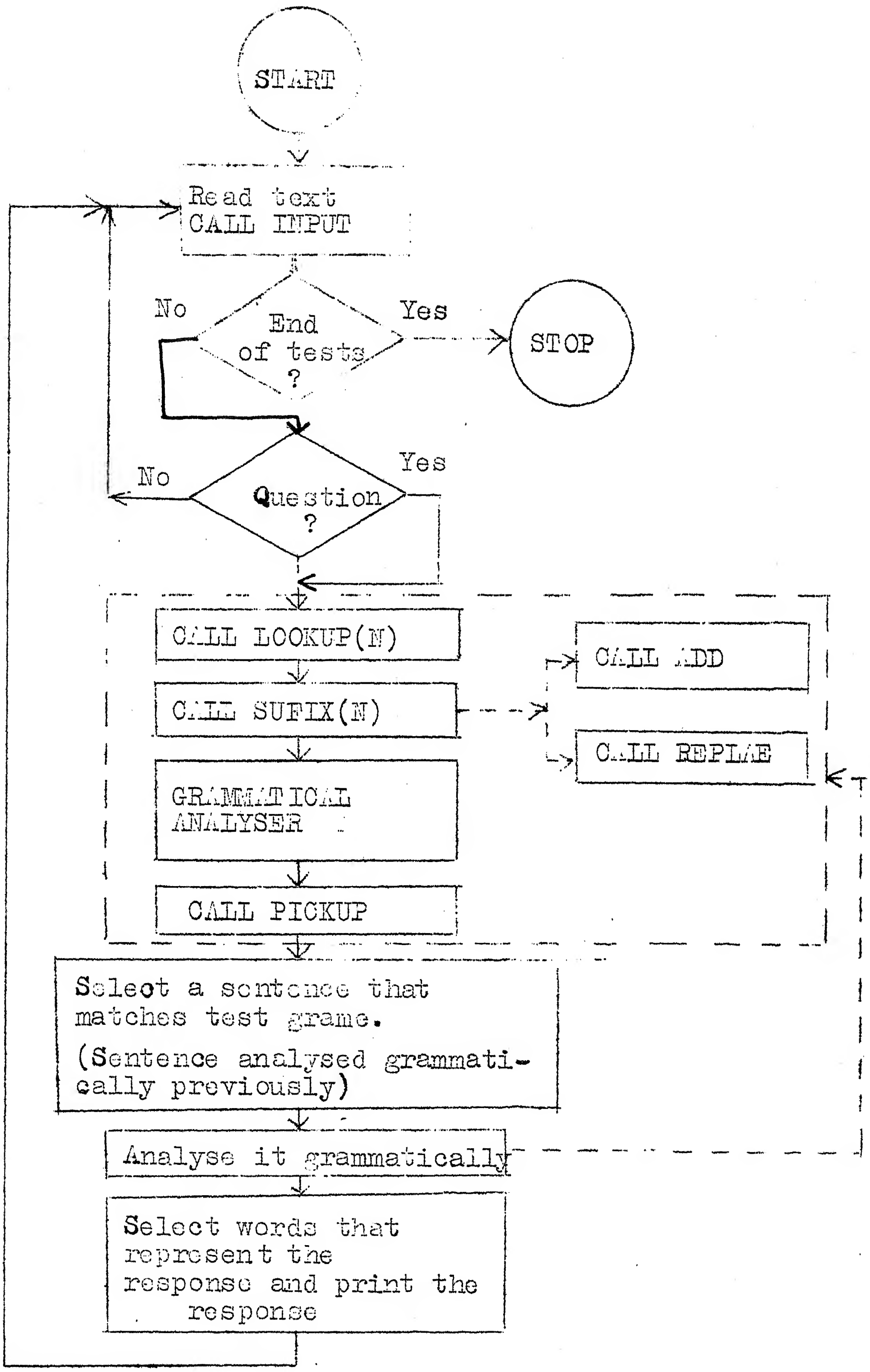
DESCRIPTION OF PROGRAM ROUTINES

In this chapter, we shall describe the individual routines through which the programmed text data undergoes grammatical analysis and processing before arriving at the guessed response for the question sentence in the text. The description of the routines is given with the help of individual flow charts. The description, throughout this chapter, assumes that the user is well aware of the grammatical units and the scheme for a detailed grammatical analysis[5] of the input text data. However, the description includes the routines for the preliminary grammatical analysis, which simplifies the task of the grammatical analyser. Before going into the individual routines, an overall flow chart for the programmed text processing is explained below.

3.1 COMPLETE FLOW CHART FOR PROGRAMMED TEXT DATA PROCESSING

The flow chart given in Figure 3.1 shows the order in which the various routines are executed, in processing the text data.

The program reads the text data till a question sentence, with one of the words as a blank entry, is present in the input. Routine INPUT enables the reading



--→ represents the call to an internal routine.

COMPLETE FLOW CHART FOR PROGRAMMED TEXT PROCESSING

of the text. As the words of an input sentence are read in, they are also searched for a dictionary match, through LOOKUP routine, in an attempt to assign a grammar code(s) to them. The presence of the question sentence (it is the last sentence in the text) calls a SUFIX routine, to assign grammatical codes to the compound words (words with suffixes) in the question sentence. Words, that have multiple grammar codes or no grammar codes are evaluated by calling the GRAMMATICAL ANALYSER routines[5]. The coded question sentence, then, picks out from the text, a sentence which is likely to contain the desired response. The selected sentence also goes through the LOOKUP, SUFIX and GRAMMATICAL ANALYSER routines, like the question sentence, and gets the grammatical codes assigned. A call to PICKUP routine, fixes the group frame of the selected sentence, which is further analysed to pick the desired response. Finally the response is printed. After all pointers are initialised, the program goes to process another text input to guess the response to its question sentence. The presence of '$$$' as the first input word of a sentence, terminates the processor program.

An overview of the dictionary and suffix organisation and an alphabetised list of variables used in the routines, given in the following three sections completes the formal introduction before the description of routines is presented.

3.2 DICTIONARY ORGANISATION

The dictionary is the main source with which the program analyses the input text at two states, (i) its primary form as soon as the text words are read in, (ii) after the text words undergo a suffix analysis, in an attempt to fix the grammatical units of the text words. It is a 1000 word dictionary built with words of common occurence in elementary English texts. It also includes some words picked from the programmed text[4] under consideration. A "frequency of occurences" analysis of the common words, in the text under consideration, has enabled the entry of the words presently used in the dictionary. Some more entries for the dictionary are also collected by running the text against the preliminary dictionary developed above and then updating it with words that have not found an entry previously on the basis of frequency analysis. Each entry in the dictionary is provided with 6 computer words, the first three for storing the word itself, the fourth for the grammatical code and fifth and sixth unused (can be used if additional information is to be stored for later processing). The dictionary lookup and retrieval of entries or updating the dictionary are simplified by the alphabetised arrangement of entries. A card at the beginning gives the count of the number of entries beginning in each of the 26 alphabetic letters. To locate a given word in the dictionary, a look at the count for the alphabet with which the word begins, pinpoints the area of scan in the

dictionary. Since there are only few entries in this area of scan, only few comparisons are needed to get a match for the given word. Similarly an altering of the count for an alphabet, enables introduction of cards with additional entries beginning with that alphabet. Multiple codes to dictionary entries are packed in a single word to save core space. Appendix C gives the dictionary organisation.

3.3 SUFFIXES AND THEIR ACTION CODES

Appendix B provides the suffixes used and their action codes. There are totally 52 suffixes employed.Since the number of characters in each are different, to provide a systematic suffix test, the suffixes are arranged in the descending order of the number of characters in them, starting from suffixes of 6 characters to those of single character. Each entry in the SUFIX1 array,used for storing the suffixes, is provided with two computer words. The first word stores the suffix. The second word is partitioned into two portions, the left two bytes (12 bits) are used for the grammar code and the right 4 bytes (24 bits) are used for action codes. A suitable mask word is used to pick the two codes separately. Each suffix is provided with one or more of the 14 grammar codes and one or more of the 8 action codes, each code specifying a suitable operation such as deletion or addition of characters to the ROOT of the word after the suffix is removed.

For example,

1. For suffix 'ING', the SUFIX1 table contains 'INGbbb' in the first word and '12bbb1' in the second word. When a word whose suffix matches 'ING', occurs in the input sentence, it is immediately assigned the grammar code of 'Form class 1 or 2' (left 2 bytes of second word). From the last 4 bytes, the action specified is 'code 1', which means 'no action' on the root of the given word, obtained after removing its suffix 'ING'.
2. For suffix 'ED', the entries are 'EDbbbb' and '23bb41'. The word with suffix 'ED' is assigned a code '23'. The action codes are '4' and '1'. Action code '1' specifies 'no action', i.e., the root of the word is to be searched for a match in the dictionary. If the search fails, action code '4' is tried which specifies 'delete last character from the root' and then attempt dictionary lookup.

From the above two examples, it is evident that the action codes are provided only for suffixes with multiple grammar code assignment, so that the action codes help in resolving the multiple assignment to unique assignment.

3.4 FORTRAN VARIABLES AND THEIR MEANING

A beforehand list of important FORTRAN variables used and their functions, given below, lessens the task of describing the routines and the drawing of the flow

charts and also gives the user a better understanding of the procedures discussed.

ALBETS(26,2)	Array for storing the alphabets in the first word and the count of number of cards that contain the entries beginning with that alphabet in the second word. By this, introducing of new words become easy. For example, if a word 'PROBABILITY' is to be introduced in the dictionary, add a card with this word just after the last word in the alphabet 'P' group and increase the count ALBETS(16,2) by 1. Later the second word for each alphabet is altered into entry pointer in the dictionary at which the alphabet starts. With this type of organisation, the boundary limits of scan for a word in the dictionary is accurately determined. For example, for a word 'STUDENT', the area of scan is given by ALBETS(19,2) to ALBETS(20,2)-1, 19 being the row number for alphabet 'S' and 20, the row number for alphabet 'T'.
ARRAY(10,2)	Stores the demarking pointers of each sentence in SEARCH by using the values of IRECOR at the beginning and at the end of each sentence as it is read in. Assumes a maximum of 10 sentences per text.

BEGN(3)	Stores the word next to '≠' entry in the question sentence. This information is useful for analysis when the possible grammar code of '≠' entry is 'Form class 3'.
CHECK(10)	Stores the grammar code of the frame of words in the group in which '≠' is a member. We call this "the group frame" of the '≠' entry. Assumes that each group frame does not have more than 10 words.
CHK	Stores the col 1 of the input data card. If it is a '*', another data card is read, taking the previous one as a comment card.
CHRCNT	Character count for an input word. CHRCNT.GT.18 prints an error message but proceeds processing ignoring the excess characters till a blank or comma is present in the input.
DCTNRY(1000,4)	Array for storing the 1000 word dictionary. The dictionary organisation has been dealt in detail above.
DOLLAR	Stores the character '$$$bbb'. An input sentence with the first word as '$$$', is used to terminate the program for end of all input texts.
ENDING	Stores the suffix of certain length (number of characters) that has been removed from the current word.

FRAME(20,4) Stores the "test frame" which consists of words and their codes collected by scanning 10 words on either side of '≠' entry (those with a grammar code of 'Form class 1' or 'Form class 2' or 'Form class 3', are chosen).

GROUPL(25,2) Stores the lower and the upper limits of the group frames in a sentence. A "group frame" in a sentence is a group formed by a frame of words in which the first word has a code of 'unit A' or 'Unit B', or 'unit F' and the last word has a code of 'Form class 1' or 'Form class 2' or 'Form class 1' respectively. Assumes a maximum of 25 groups for each sentence.

ICOUNT Pointer to the number of entries in WORD. A value greater than 51 prints an error message and stops processing.

IEMP(9) A single dimensioned array of nine words that contain the characters, 'E', 'Y', 'ITY', 'D', 'IDE', 'OF', 'LE', 'LA'. Any one or more than one of these characters need to be added to the ROOT of the word (after its original suffix has been removed) before a dictionary lookup is effected. This array is mainly of use in SUFIX test.

IMARK(=HASHF)	Stores the special character '≠', to be used in the blank space of the question sentence.
INP	Linear pointer that scans a data card read characterwise. INP.GT.65 signals the end of a data card.
IRECOR	Pointer to number of words stored in the SEARCH array. IRECOR.GT.50*10 prints an error message and resets all pointers for a fresh text reading.
IVALUE	Stores the grammar code of the dictionary entry that has matched the root of the word.
JSNT	Number of sentences read into SEARCH from the input text.
K2	Number of characters in the ROOT of the word.
KNPP	Pointer to the number of groups present in the sentence.
LIM	Pointer to the number of entries in the dictionary.
LINE(65)	Contents of a single data card from col 2 to col 66 are read into this array.
LLK	Pointer to number of possible responses to the question sentence.
LOC	Stores the possible grammar code of the '≠' entry in the sentence.

LOCAL	Temporary location for storing input data words before restoring them to WORD.
LPTR	Pointer to number of frame words in the test frame.
ROOT(3)	Stores the root of the word after the suffix has been removed.
SEARCH(500,6)	Stores the input text of sentences one by one wordwise. This provides an effective storage of input text of a maximum of 10 sentences, each sentence being 50 words long.
STORE(200,5)	Array that provides the output. The first three words for each entry stores the possible answer to '≠' entry, the fourth its grammar code and the fifth, the selected sentence in the text from which that entry is picked out. Assumes a response of 20 words from each sentence (maximum).
SUFIX1(52,2)	Array for storing the suffixes and their grammar codes and action codes. The organisation is given in section 3.3.
SWITCH	A variable that is assigned one of the four values, 1,2,3,4 to flag the input card read, at appropriate places. SWITCH = 1 is assigned at the beginning of an input sentence. It remains in that value until a character other than a blank or comma or period occurs.

SWITCH = 2 is assigned when an alphabetic character is encountered thus flagging the beginning of a word. It remains in that value as long as only alphabetic characters are present in the input word.

SWITCH=3 flags the presence of an alpha-numeric or special character in a word, thus suppressing the dictionary lookup and subsequent suffix test for such entries. It remains in that value, so long as alphanumeric or special characters arrive successively at the input.

SWITCH=4 is assigned when the end of an input word is reached. It remains in this value till a new word is encountered (when its switches to 3 or 2) or end of sentence is reached (indicated by a period).

TAB(7) — Stores the pointers in the reverse order, that demark the different suffix groups classified according to the number of characters in them. TAB is used to fix the limits of the scan of SUFIX1 for a match to the given sufix.

WORD(51,6) — Array for storing the sentence wordwise. Provides a maximum of 51 words for each sentence stored. Each entry in WORD has three

three computer words for storing the entry, fourth for storing its grammar code fifth and sixth unused (can be used to store other information like entry's suffix, its grammar code etc.).

Apart from these variables, few logical variables employed are described in the routines in which they find their place.

3.5 DESCRIPTION OF ROUTINES AND FLOW CHARTS

Any natural language data processor for solving programmed instruction text consists mainly of two steps:

(a) grammatical coding of input text

(b) the actual processing using the above coding to guess the response for the blank entry at the end of each text.

The explanation of the routines used for the grammatical coding of input text is not of interst in this report. However, to make an efficient use of the grammatical analysis, few routines like dictionary lookup, suffix testing are provided before giving the text as input to the grammatical processor. The organisation of the routines is

(1) INPUT of the programmed text

(2) dictionary LOOKUP

(3) SUFIX testing

(4) ADD routine

(5) REPLAE routine

(6) PICKUP routine

(7) MAIN routine.

3.5.1 INPUT Routine

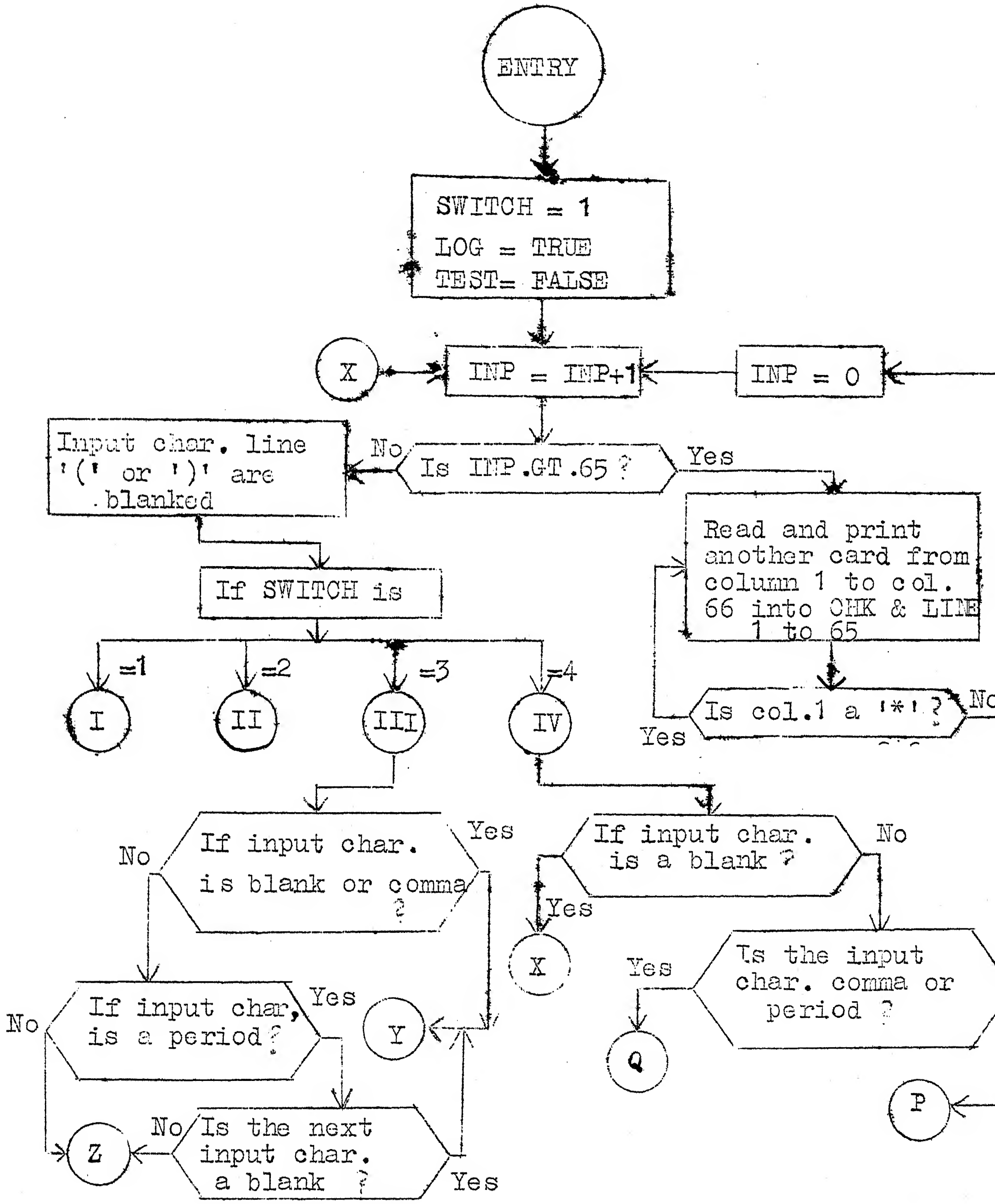
This routine is intended for reading in the programmed text data sentencewise. It also prints out the sentences as they are read. It also provides a simultaneous lookup of words, as they are read in, through a call to LOOKUP routine. It, however, assigns an indicator to the blank entry and to alphanumeric words in the input text thus suppressing an otherwise made dictionary lookup with no success. A logical variable TEST is turned .TRUE. to indicate to the MAIN routine that the input text is over (i.e., the sentence with '≠' entry has been read) and it can proceed to process the text. A variable SWITCH facilitates storing an input sentence wordwise into WORD. The presence of a period followed by a blank or an alphabetic letter signals the end of a sentence and returns control to MAIN. Attempt has been made to provide the user a free format of giving the text but to stick to the nature of ordinary English texts one has to remember a few points.

(1) Input text is given in cards from col 2 to 66.

(2) A '*' punch in col 1 treats the card as a comment card.

(3) Each text word not exceeding 18 characters should be delemited with a blank or a comma.

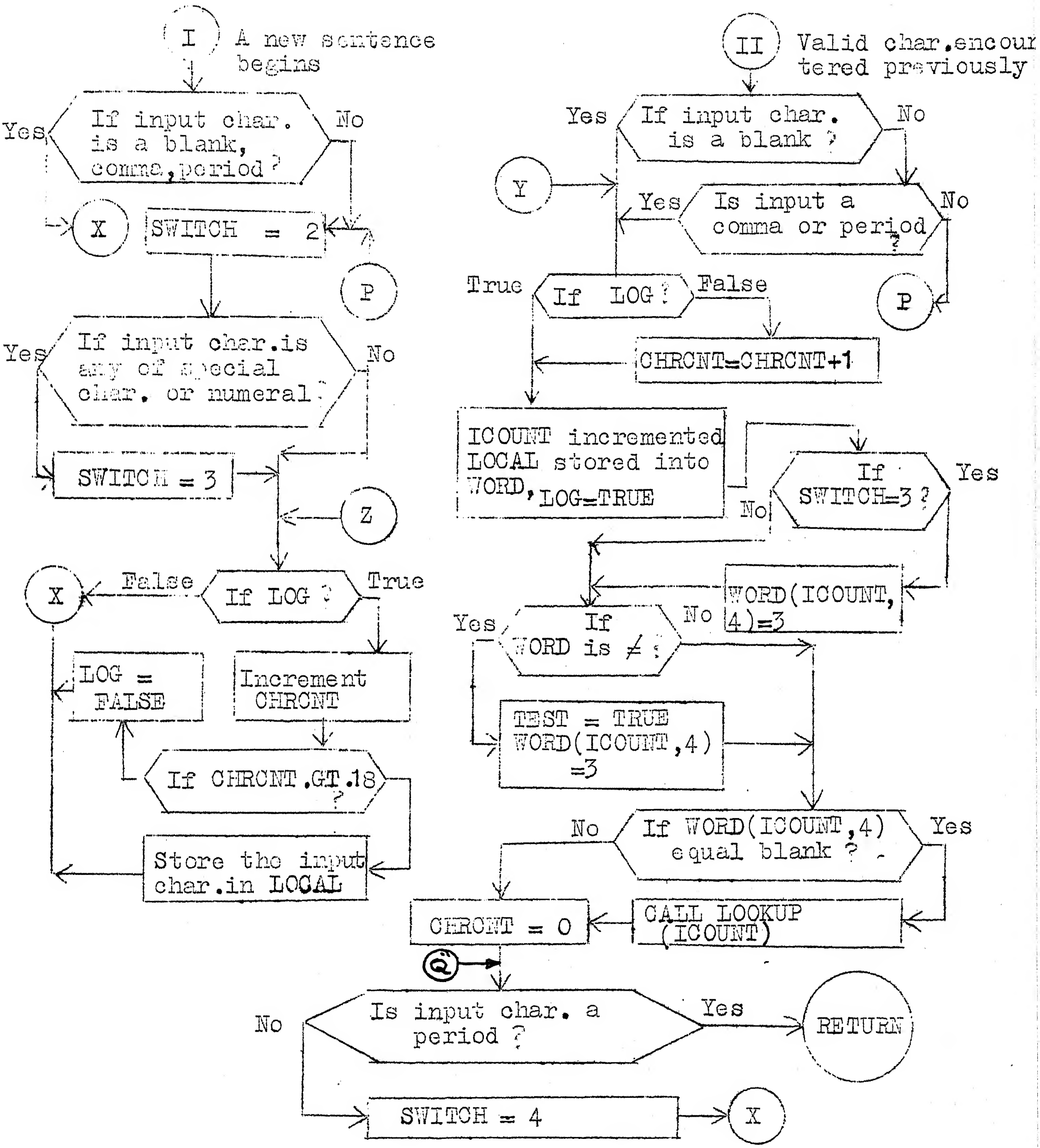
(4) Successive blanks or commas or periods in the input text are ignored.



Stage III : Input word has a special character or numerical character (assigned grammar code is integer value 3).

Stage IV : Read a blank or camma in the input text (a word has just been stored into array 'WORD')

FLOW CHART 'INPUT' (CONTD.)



FLOW CHART 'INPUT'

(5) A period is used to signal the end of a sentence. However, a period between two numeric character is treated as the decimal point.

(6) Words with alphanumeric characters are specially tagged.

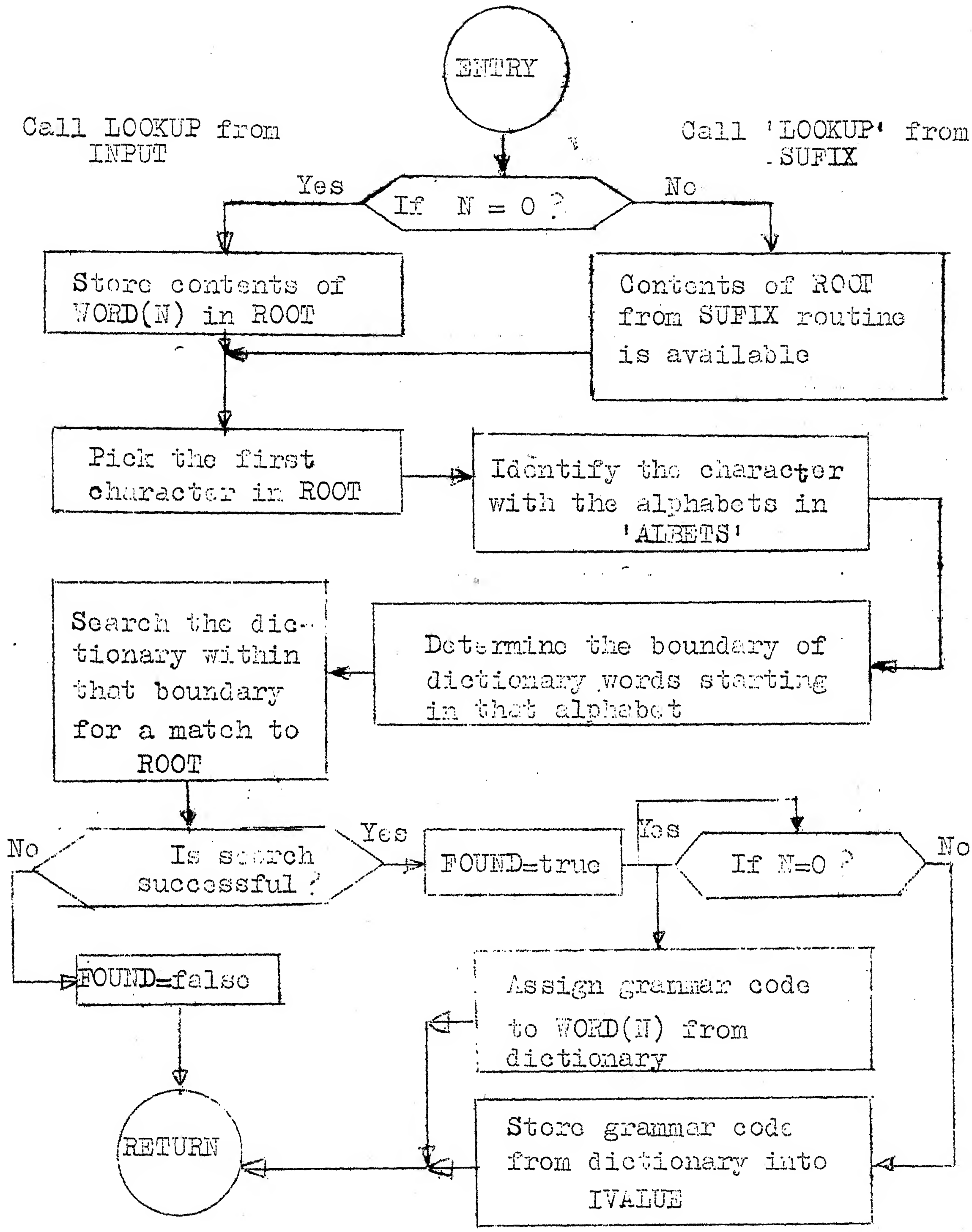
(7) No more than 50 words are allowed for a sentence and no more than 10 sentences for each text.

(8) A sentence starting in '$$$' as the first word signals the end of all input texts.

For more details refer to the flow chart given in Figure 3.2

3.5.2 LOOKUP Routine

Each word read from the input text through INPUT routine is searched in the Dictionary in an attempt to assign a code to the word. A code is assigned if a match is found in the dictionary. A similar call made from SUFIX routine helps to find a match for the ROOT of the word in the dictionary. The SUFIX routine requires a dictionary lookup after the execution of each action code of the suffix satisfied. The pre-assignment of grammar code to known words is done in order to simplify task of the grammatical analyser. The argument N gives the entry number in WORD if a lookup is done from INPUT. The argument is zero if a call is from SUFIX routine. This is because in one case WORD(N) is searched and in the other case the ROOT is searched. Searching for a match in the dictionary consists



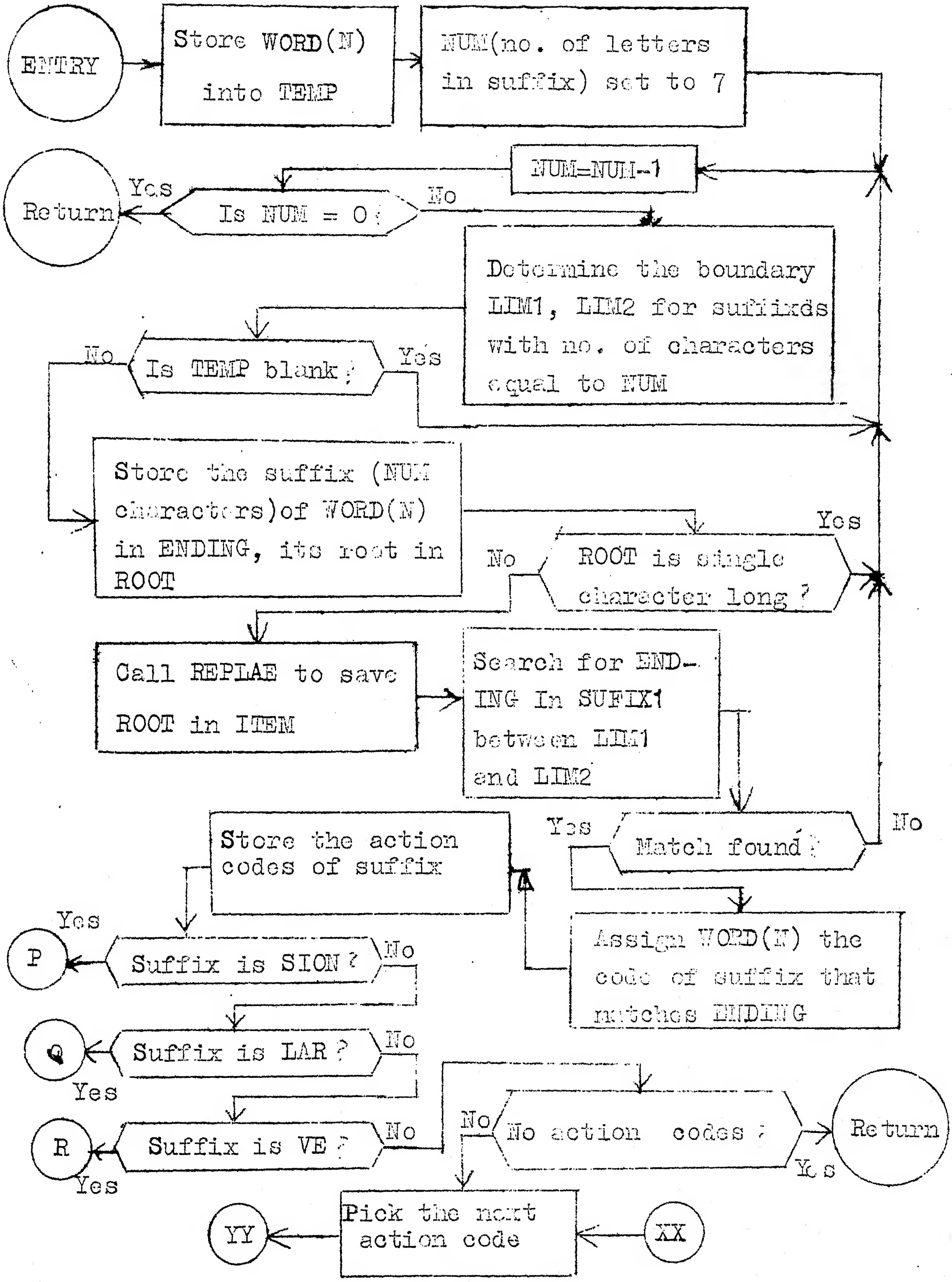
FLOW CHART FOR 'LOOKUP(N)' ROUTINE

N - entry in WORD which is searched.

in determining the starting alphabet of WORD or ROOT and identifying it with ALBETS which gives the lower and upper limits of the scan. A logical variable FOUND is used to indicate whether the scan is a successful one or not.

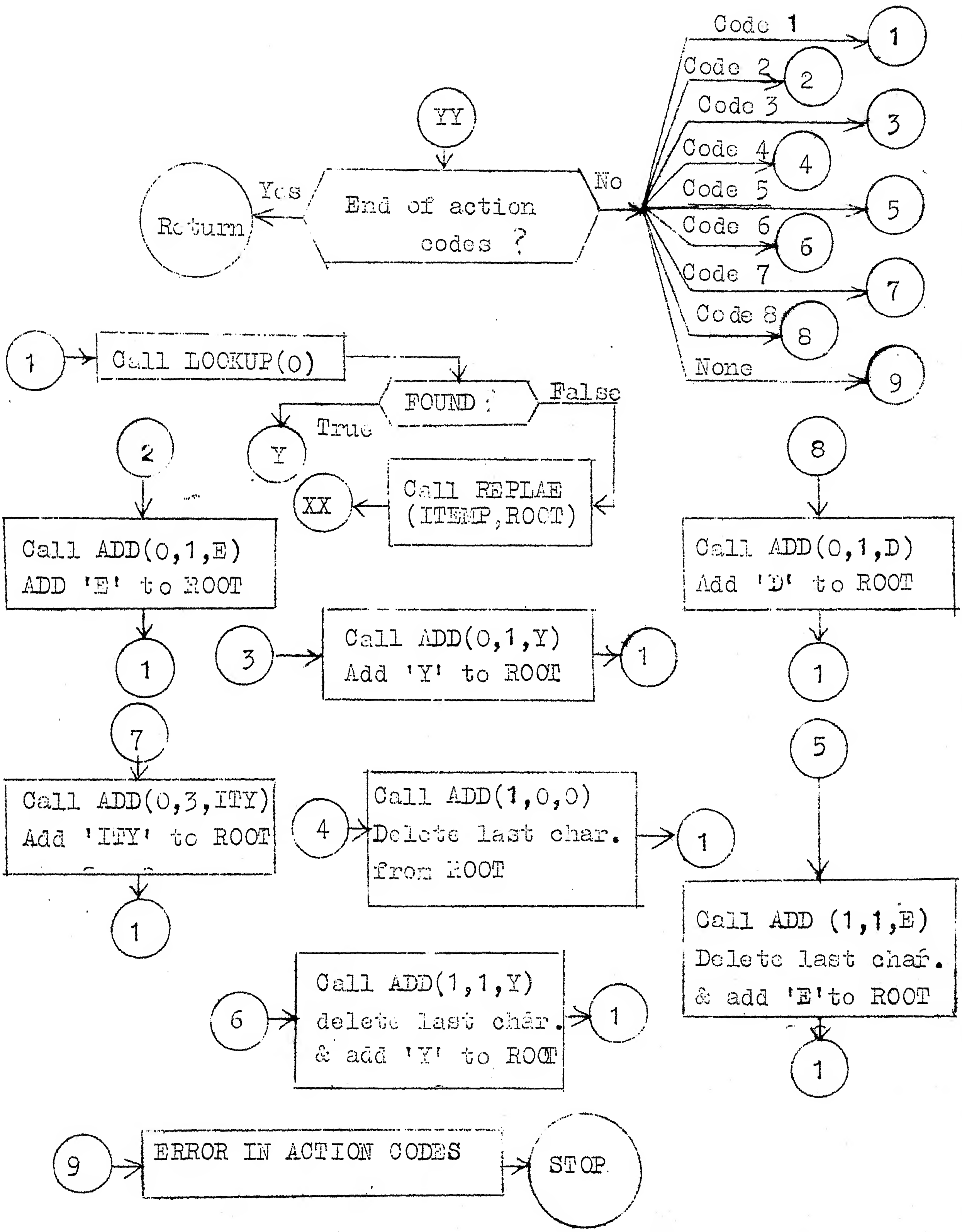
3.5.3 SUFIX Routine

English word in most cases occur with a suffix, especially when one deals with plurals of nouns, past tense of verbs etc. Due to the restriction of using a limited class of dictionary words, the dictionary does not provide suffixed words. To overcome this, one considers a SUFIX routine to perform the operation of removing the suffix, identifying it with standard suffixes previously stored and assigning to the word the grammar code of the suffix if a match is found. In addition to it, the SUFIX routine performs some additional operations like deleting or adding character(s), to the ROOT of the word depending on the action code(s) of the suffix matched. It also does a dictionary lookup of ROOT through LOOKUP routine. If a dictionary match is found, the intersection of the two codes, one provided by suffix match and the other by the dictionary, is assigned as the grammar code for the present word, WORD(N), for which the test is carried out. A systematic manner of suffix test, starting from the first group of suffixes (suffixes of 6 characters) to the last group (suffixes of single character), in most cases, enables one to arrive at the actual suffix of the

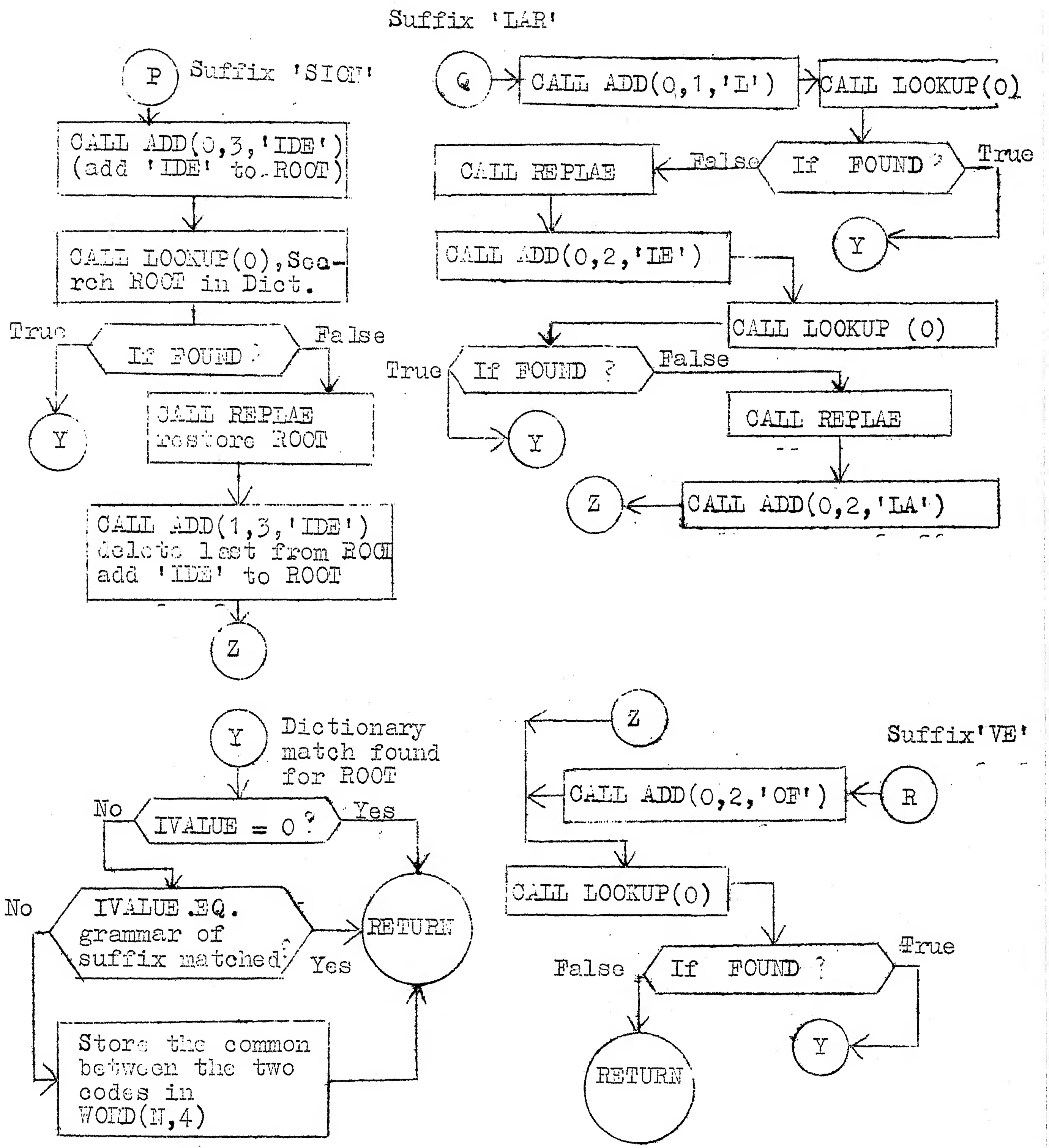


FLOW CHART FOR 'SUFIX(N)' ROUTINE (CONTD.)

N - entry in WORD which is searched.



FLOW CHART FOR 'SUFIX(N)' ROUTINE (CONTD.)



FLOW CHART 'SUFIX(N)'

word. Since, for each action code of a suffix, the ROOT undergoes modification for dictionary lookup, a routine REPLAE is called to save and later on restore back the contents of ROOT for further analysis. Another routine ADD is called to perform addition or deletion of characters from ROOT as required by the suffix. Just like the LOOKUP routine, the SUFIX routine also simplifies the task of the grammatical analyser. Inspite of the grammatical analysis using dictionary lookup and suffix, it has turned out that some input words may not yet have grammatical units assigned. A heuristic scheme developed by (5) has to be adopted in such cases. However, the suffixes and actions that have been used, provide more or less a general scheme for evaluating the grammar of the word.

3.4.4 REPLAE Routine

This is a small routine intended to store the contents of the first argument in the second argument. This routine is called from SUFIX routine. Each time an action code of a suffix is executed contents of ROOT get modified by addition or deletion of characters to or from it. One feels safe if the modified ROOT finds a dictionary match, but in case it fails, one has to restore the original ROOT of the word and apply next action code if there is one. To achieve this, one calls REPLAE routine before the execution of the action code, to save contents of ROOT in ITEMP. After the code is executed, one again calls the REPLAE routine to restore the contents of ROOT from the temporary location ITEMP.

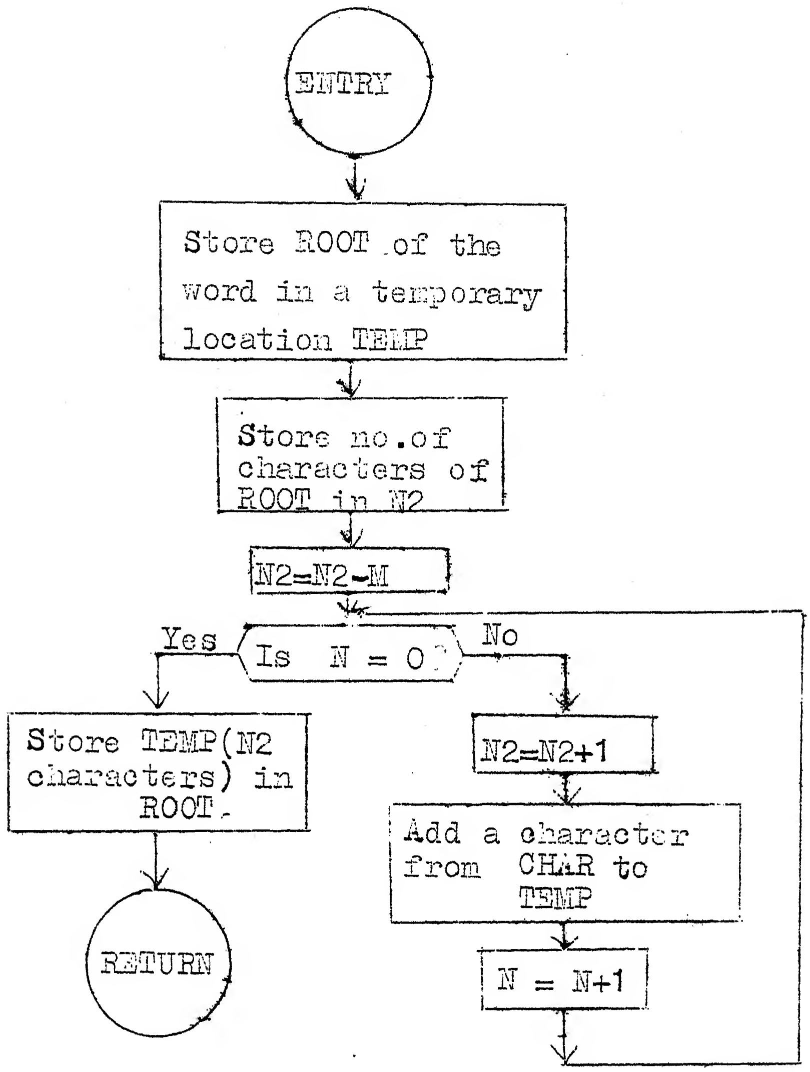
3.4.5 ADD Routine

The action codes of a suffix demand certain number of characters to be deleted from or added to the ROOT of the word. This addition or deletion becomes necessary because when some basic words are used with the suffixes, their root might have been changed a little to form compound words with the suffixes. This routine facilitates simeltaneous deletion and addition of characters in the ROOT. Out of the three arguments used, the first one 'M' specifies the number of characters to be deleted, the third 'CHAR' and the second 'N' specify the word to be added and the number of characters in that word. Since N never exceeds a value of 3, the three computer words for the ROOT is still sufficient when it is modified.

3.4.6 PICKUP Routine

A pickup routine has been provided to group the words in a sentence according to selected grammar frames. This assumes that the sentence has already undergone an elaborate grammatical analysis and that each word is provided with a unique grammar code as far as possible.

The routine scans the sentence, looking into the grammar category of each word, forms a group if a grammar frame starting in 'functional unit A' and ending in 'Form class 1' is met. Similarly it proceeds scanning and forms a group whenever a 'B-2' frame or a 'F-1' frame or frames

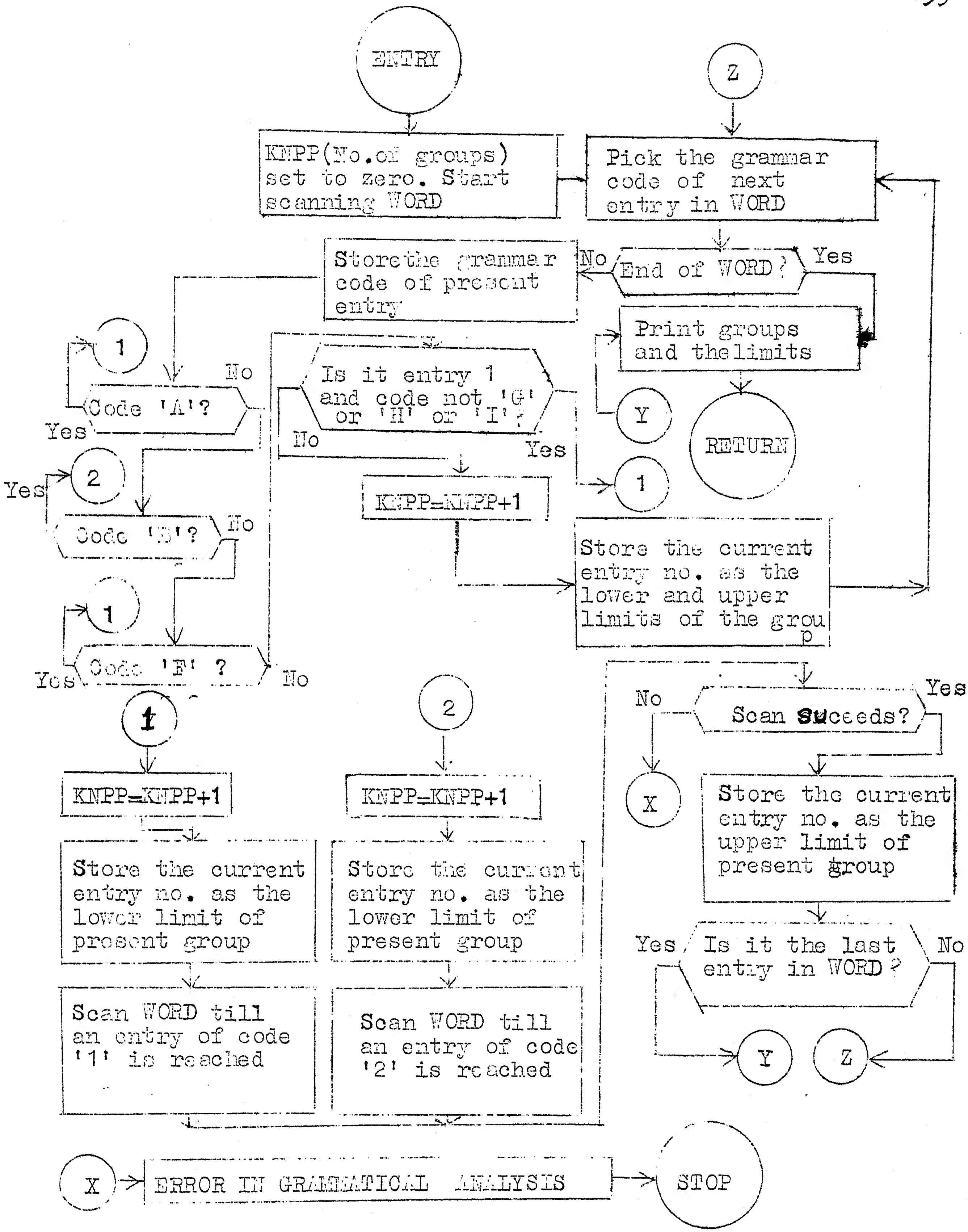


FLOW CHART FOR 'ADD(M,N,CHAR)' ROUTINE

M - no. of letters to be deleted from ROOT

N - no. of letters (in CHAR) to be added to ROOT

CHAR - contains characters to be added to ROOT.



FLOW CHART FOR 'PICKUP' ROUTINE

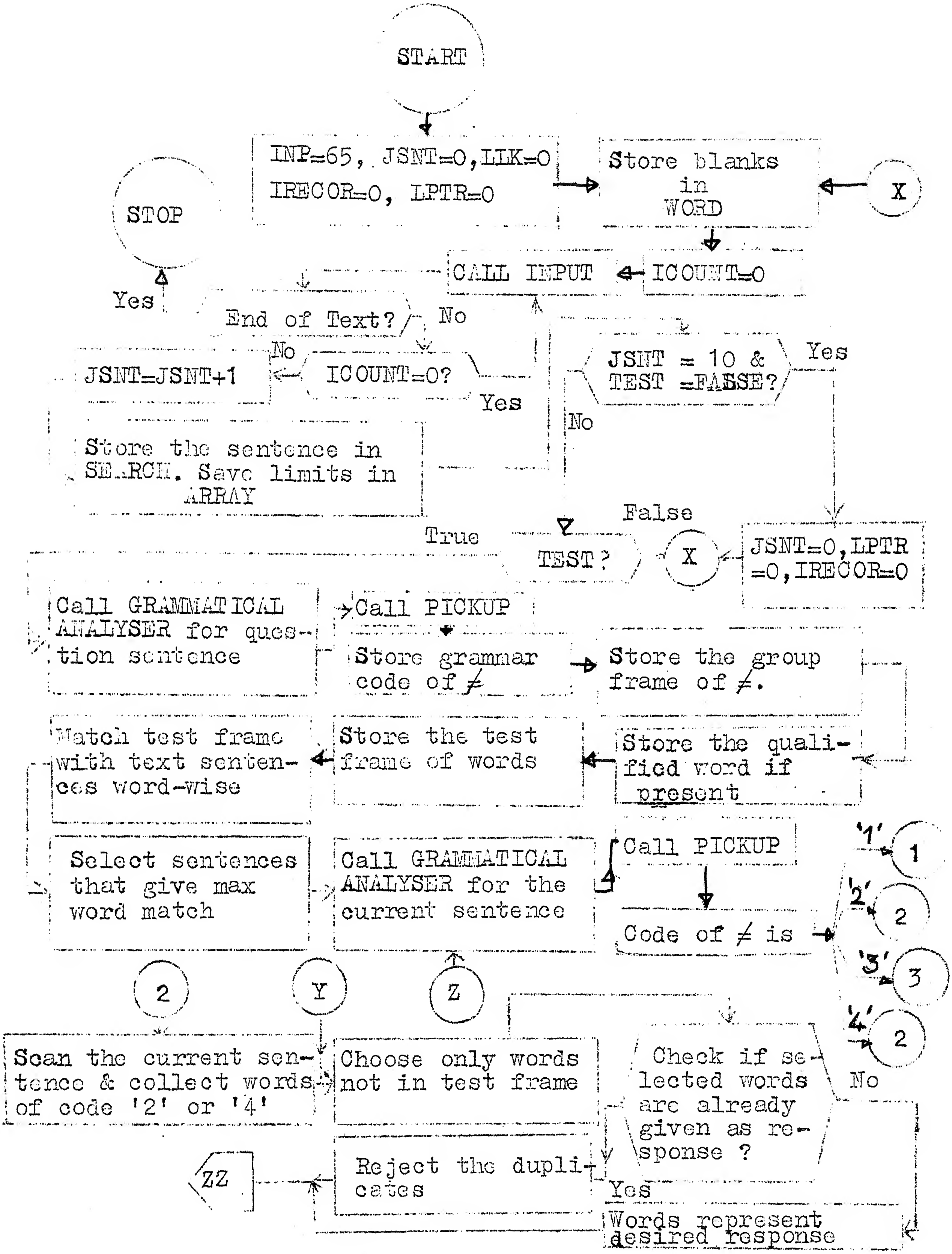
not belonging to any of these is met. This routine returns the groups and their limits in the sentence as its value. This group selection has been done with a view to accurately pickout the correct word from the sentence. This is done by comparing such group frames of the selected sentence with the group frame of the blank entry in the question sentence. This analysis produces successful results when the blank entry is possibly a noun with a qualifier before it or it is qualifying a noun after it. By comparing the qualifiers in the two group frames one picks out fairly correct word required.

3.4.7 MAIN Routine

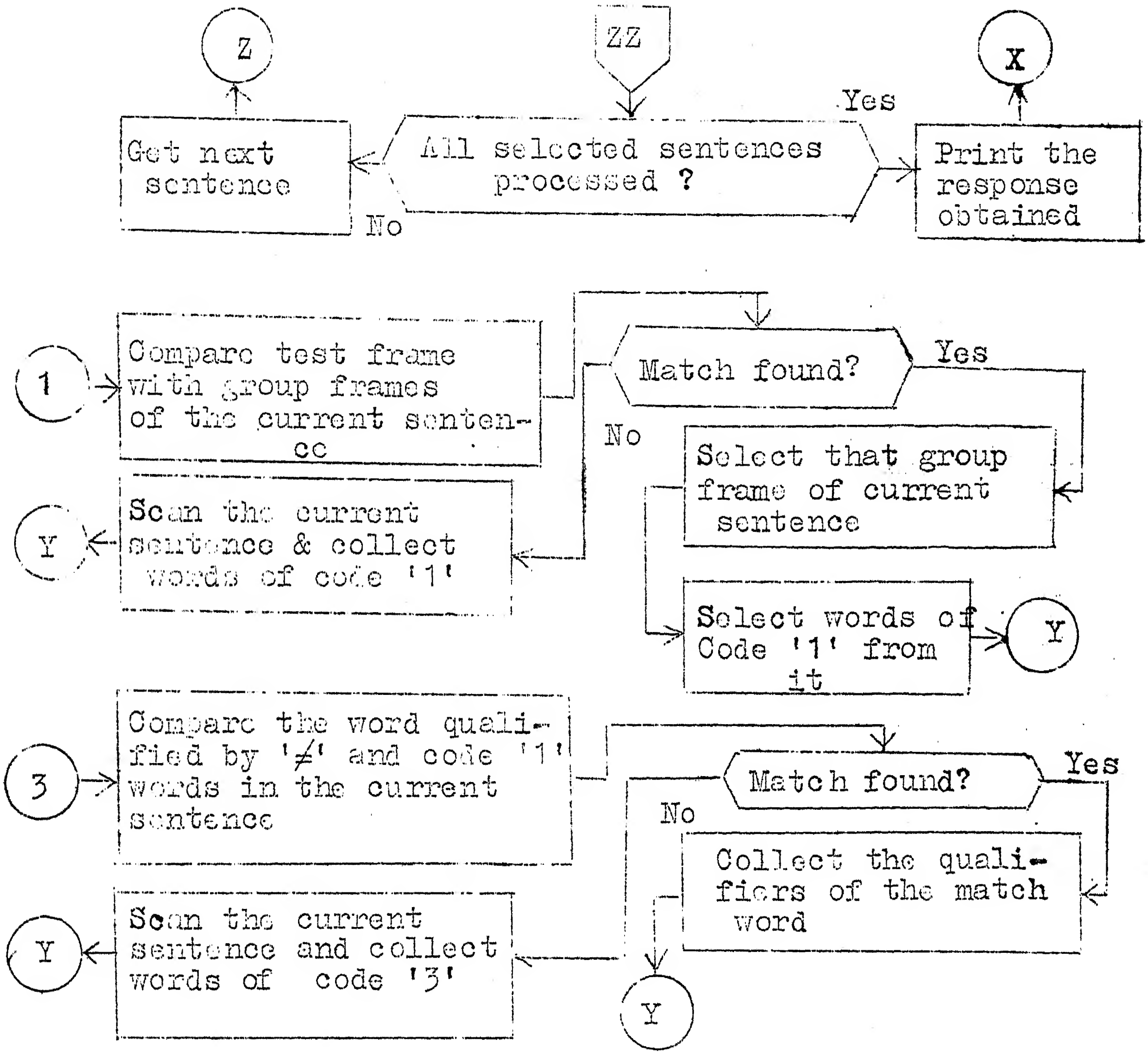
A slight deviation in the description of routines is done by providing the MAIN at the end. The main routine after calling in the various subroutines discussed above, analyses the results obtained in doing the text processing. The flow chart for the MAIN has been given a greater detail to reduce the content of description here. The essentials of the routine are however given below for completion. The sequence of events in the processing are,

(1) Read in the programmed text through INPUT routine.

(2) Get the sentence with the blank entry grammatically coded. Collect the group frame and the grammar code for the '≠' entry in the sentence. Collect the "test frame by scanning on either side of the '≠' entry. Store the word next to'≠ 'if possible.



FLOW CHART FOR 'MAIN' ROUTINE (CONTD.)



FLOW CHART FOR 'MAIN' ROUTINE

(3) By a context matching using the test frame, pick the sentences from the text that provide maximal match. Get the grammatical structure of these sentences.

(4) Since the grammar code of the '≠' entry will be one of the four 'Form class words', do the analysis separately.

(a) If the code of the '≠' entry is 'class 1' then the group frame which contains the '≠' entry is matched against the group frames of the selected sentence. If the two group frames possess atleast one matched grammatical unit, then that group frame of the selected sentence is chosen. The words with grammar code of'class 1' in the selected group frame are given as the response.

(b) If the grammar code of '≠' is 'class 2' or 'class 4' all the words in the selected sentence having the grammar code of 'clas 2' or 'class 4' are selected as the response.

(c) If the grammar code of '≠' is 'class 3' a qualified word('class 1' word) follows it. 'Class 1' words in each group frame of the selected sentence are matched against the qualified word mentioned above. If a match occurs the qualifiers preceeding these 'class 1' words in the group frames of the selected sentence are given as the response. In all above cases, if a match does not occur, the

words having the same grammar code as '$\neq$' entry are picked from the selected sentence.

(5) The final phase is the presentation of the output in a suitable format.

Although the MAIN routine does an elaborate analysis to arrive at a unique response, still the output takes accuracy and presents ambiguity because of the absence of precise information about the grammar of the text processed. The MAIN also provides an error message, "MORE THAN 10 SENTENCES IN A TEXT, ALL POINTERS INITIALISED" when a text containing more than 10 sentences (maximum limit due to core restriction) is read in. An error message, "DIMENSION OVERFLOW IN XXXXXX ARRAY" is used to check any array exceeding its maximum dimension.

The next chapter discusses some aspects of the results obtained. It also points out some limitation of the scheme and suggestion for improvement and extension of the scheme to advanced fields of text processing.

CHAPTER IV

RESULTS AND CONCLUSIONS

4.1 RESULTS

A test run, carried out on a few passages picked from a programmed instruction manual for teaching programming techniques[4] is presented in Appendix A. The output is printed out at various stages of analysis of the programmed text data by the processor program. The stages of output are:

(i) after the text containing the question sentence has been read.

(ii) after the grammatical analysis of the question sentence in the text by the grammatical processor.

(iii) after the test frame of words is collected from the question sentence

(iv) after picking and grammatically analysing the sentence, which is likely to give the desired response.

(v) after determining the group frames for the selected sentence and the group frame for the blank entry in the question sentence.

(vi) finally, after picking out the possible response(s) for the programmed text from the selected sentence.

Although developed on a heuristic notion, the scheme has, on many programmed text inputs, given good and more or less accurate responses. In some cases only, the responses obtained have turned out to be ambiguous. To enable the user to detect the possible punching mistakes in the text data, the program prints out error messages, which are themselves self-explanatory. For example

1. "ERROR-NUMBER OF CHARACTERS IN A WORD EXCEEDS 18"
   This error might have arisen either due to the missing blanks between successive words or due to the programmer's intention to use lengthy words. In this case, the program prints the error message, ignores the characters that occur after the maximum limit until a blank or a comma is read, which signals the end of that word. The word in error is also printed along with the error message.
2. "ERROR-MORE THAN 50 WORDS IN A SENTENCE"
   This error might be due to a missing period between successive sentences or might be due to the users intention to use lengthy sentences. The program prints the error message and stops processing. The sentence in error is also notified in the printout.

A few error messages like, "ERROR IN SUFIX ANALYSIS", "ERROR IN GRAMMATICAL ANALYSIS", "ERROR IN ACTION CODES" etc. have been provided to detect the malfunctioning of the system while executing the program. Appendix B provides a list of

suffixes and their action codes. Appendix C provides the 1000 word dictionary. Appendix D gives the program listing.

4.2 CONCLUSIONS

Preparation of a programmed text for any application is a difficult task. And those which have been preparéd, need to be evaluated, to determine whether they serve the purpose for which they are intended. By having a scheme for programmed text processing, one gets an insight into the text to decide whether it is a "bad one" (not well prepared) or "a good one" (well prepared).

A look at the nature of the output obtained for our scheme points out, that in some cases, ambiguous words (sometimes totally wrong words) have been guessed as the possible response to the question sentence in the text. This means that the text and the question sentences have been framed properly. Because, if a computer, provided only with the grammatical information on the text and no intelligence with regards to the meaning of the sentences in the text, could guess the response accurately in all cases, then an user of the programmed text, given the same background could also guess the response without actually understanding the contents of the text. This is not, what we desire from an user. By having ambiguous responses, the task of a programmed text user is made difficult in guessing the correct response, because he must now understand the text first and then attempt to answer the question sentence. This above facility provided by the scheme given in this report, aids in the preparation and the evaluation of programmed texts with grammatical background alone, which is what we aimed at doing.

The present dictionary could be developed into a "thesaurus" type to incorporate additional grammatical information to each entry of the dictionary. Here each entry in the dictionary is provided with two codes: A grammar code calls in appropriate routines for grammatical analysis. The grammar code could also provide additional grammatical information, instead of the broad classification used in this report. For example, if an entry is a noun or a verb, we could add details like its gender, case, number, etc. And a semantic code for each entry could be a pointer to another small dictionary giving the antonyms, synonyms, word pairs etc. of the present entry. A "thesaurus" type of dictionary with all above mentioned informations could make more accurate text processing possible. However, one is cautioned against the development of such a dictionary, because, by introducing all words, the dictionary becomes, one that represents the background of the designer of the programmed text and not one that represents the background of the user. So when a "thesaurus" is constructed, it should cover only the background of the student, who is going to use the text. The "thesaurus" could, however, be updated when it has to deal with programmed texts, prepared for different applications. When a "thesaurus" is to be used, more core space might be occupied by the dictionary itself, leaving only a little space for the program and the data text. If, the quantity of text processed and not the text processing speed is of importance,

the scheme could make use of backups like discs, tapes etc. for storing the dictionary and the programmed text data. Routines should be included in the present scheme, to take care of this.

Since the test frame, described in the report, is the one that picks the sentence, which is most likely to contain the desired response, one would like to have additional words of importance in it. At present only 10 words on either side of the blank entry in the question sentence, form the test frame. Instead, the whole sentence could be scanned and words of specific grammar category could be picked or if special punctuation marks are present in the sentence, they could be made use of in splitting the sentence to distinct parts and analysing the parts individually.

Another development of the scheme, is to take care of multiple blank entries in the question sentence. One suggestion is to provide as many passes of the text as the number of blank entries, the results of each pass being used in subsequent passes. This may not, however, ensure accurate analysis and also any wrong response to the first blank affects the response to subsequent blanks. It is also possible, in a complete programmed text, such question sentences might appear frequently between successive passages of the text. The scheme could be extended to take care of such cases also, by providing a more precise grammatical analysis and an accurate context matching. One should not

lose track of the context, the results, the frame words and other informations used in guessing the response to the first question sentence. This is because, as we move to successive passages of the text, informations obtained from the previous passage may have a relevance in guessing the response to the question that appears in the next passage.

Finally, the development of the programmed instruction texts has been one of the contributing factors to the growth of computer assited instruction (CAI), in that, they provide individualized instructions and assist authors in the development of the instruction materials. The scheme, developed in this report fulfils the first requirement of the CAI for large educational system, by framing the instruction material into a programmed instruction text with the aid of computer produced response. By incorporating the text with the correct responses in the system, an on line computer can evaluate the performance of a student, by presenting him with the text minus the responses, asking him to guess the response for the question sentences and then by comparing his responses with the correct responses and taking a statistics of the responses provided by him over a number of texts under the same subject. With the advent of time sharing systems and the capability of the central processor to maintain more than one terminal simultaneously the computer can actually conduct a CAI course[7], thus providing a continuous and automatic assessment of the students' abilities and potentials over the subject under discussion.

REFERENCES

1. Halpern,M.I., "Foundation of the Case for Natural Language Programming", IEEE Spectrum,Vol.4,No.3, March 1967, pp 140.

2. Ray,L.C.,"Programming for Natural Language", Garvin,P.L.,"A Linguistic view of Language-Data Processing",in Garvin,P.L.(Ed.),Natural Language a nd the Computer,New York,McGraw-Hill Book Co., Inc.,1963,pp 95,pp 109.

3. Harold Borko(Ed.),Automated Language Processing, (Book),New York,John Wiley and Sons Inc., 1967.

4. Brethover,D.M.,Programmed Instruction, a manual of Programming Techniques,(Book),Chicago, Educational Methods Inc., 1963.

5. Venkataraman,S.,"A Grammatical Analyser for Natural Language Texts",M.Tech.Thesis,Department of Electrical Engineering,Indian Institute of Technology,Kanpur, August 1972.

6. Emile Delavenay,An Introduction to Machine Translation ,New York,Frederick.A.Praeger Inc.,1960

7. Atkison,R.C. and Wilson,H.A.(Eds.),Computer-Assisted Instruction,New York,Academic Press,1969.

APPENDIX A

SAMPLE OUTPUT

NOTE: Numerals or alphabets within quotes represent grammatical codes.

Example 1

INPUT TEXT

TEACHING STUDENTS AND TRAINEES TO SAY AND DO CERTAIN THINGS IS THE MAJOR CONCERN OF THE NEW TECHNOLOGY PROGRAMMED INSTRUCTION. A LIST OR SPECIFICATION OF THE THINGS THE PROGRAMMER WANTS THE STUDENTS TO BE ABLE TO SAY AND DO AT THE END OF THE PROGRAM IS CALLED A SPECIFICATION OF THE TERMINAL BEHAVIOUR FOR THE PROGRAM. A SPECIFICATION OF THE TERMINAL ≠ IS IMPORTANT TO THE PROGRAMMER.

GRAMMATICAL ANALYSIS OF QUESTION SENTENCE

A	SPECIFICATION	OF	THE	TERMINAL	≠	IS	IMPORTANT
'A'	'1'	'F'	'A'	'3'	'1'	'2'	'1'

TO	THE	PROGRAMMAR
'F'	'A'	'1'

GROUP FRAMES OF QUESTION SENTENCE

1 1-2 2 3-6 3 7-7 4 8-8 5 9-11

TEST FRAMES

WORDS	CODE
SPECIFICATION	'1'
TERMINAL	'3'
IS	'2'
IMPORTANT	'1'
PROGRAMMER	'1'

SELECTED SENTENCE AND ITS CODES

```
A    LIST  OR  SPECIFICATION   OF   THE   THINGS  THE
'A'  '1'   'E' '1'             'F'  'A'   '1'     'A'
PROGRAMMER WANTS THE STUDENTS TO  BE  ABLE TO  SAY  AND
'1'        '2'   'A' '1'      'B' 'B' '2'  'B' '2'  'E'
DO  AT  THE END OF  THE PROGRAM IS  CALLED A
'2' 'F' 'A' '1' 'F' 'A' '1'     'B' '2'    'A'
SPECIFICATION OF  THE TERMINAL BEHAVIOUR FOR THE PROGRAM
'1'           'F' 'A' '3'      '1'       'F' 'A' '1'
```

GROUP FRAMES FOR THE SELECTED SENTENCE

```
1   1-2    2  3-3    3  4-4    4  5-7    5  8-9
6   10-10  7  11-12  8  13-15  9  16-17  10 18-18  11 19-19
12  20-22  13 23-24  14 26-27  15 28-29  16 30-33
17  34-36
```

COMPUTER PRODUCED RESPONSES TO THE QUESTION SENTENCE

ENTRY	CODE	SENTENCE
TERMINAL	'3'	0
BEHAVIOUR	'1'	2

Example 2

INPUT TEXT

PROGRAMED INSTRUCTION MAY USE ITEMS OF THIS TYPE. A WORD IS MISSING FROM A SENTENCE, AND THE STUDENT IS REQUIRED TO SUPPLY THE MISSING WORD. WE CAN PROGRAM A TEXT BOOK ON LOGIC. PART OF THE JOB ENTAILS WRITING SENTENCES THAT HAVE ATLEAST ONE WORD MISSING. THE STUDENT IS ASKED TO WRITE THE MISSING WORD. WE CAN ALSO PROGRAM A TEXT BOOK ON PHYSICS, GRAMMAR, ART, OR PERSONNEL MANAGEMENT. REGARDLESS OF THE SUBJECT MATTER, IF WE WRITE A PROGRAM CONSISTING OF INCOMPLETE SENTENCES, WE WOULD REQUIRE THE STUDENT TO WRITE THE MISSING WORD. WE USE A TAPE RECORDER IN PROGRAMMING A SPOKEN LANGUAGE. THE STUDENT HEARS A SENTENCE IN THE LANGUAGE HE IS TO LEARN. A PHRASE IS MISSING FROM THE SENTENCE AND THE STUDENT IS REQUIRED TO WRITE THAT ≠.

GRAMMATICAL ANALYSIS OF QUESTION SENTENCE

```
A   PHRASE IS  MISSING FROM THE SENTENCE AND THE STUDENT IS
'A' '1'    'B' '2'     'F'  'A' '1'      'E' 'A' '1'     'B'
REQUIRED TO  WRITE THAT ≠
'2'      'B' '2'   'A'  '1'
```

GROUP FRAMES OF QUESTION SENTENCE

```
1  1-2  2 3-4  3 5-7  4 8-8  5 9-10  6 11-12  7 13-14
8  15-16
```

TEST FRAMES

WORDS	CODE
SENTENCE	'1'
STUDENT	'1'
REQUIRED	'2'
WRITE	'2'

SELECTED SENTENCE AND ITS CODES

```
A   WORD  IS   MISSING FROM  A   SENTENCE AND THE STUDENT  IS
'A' '1'   'B'  '2'     'F'   'A' '1'      'E' 'A' '1'      'B'
REQUIRED TO SUPPLY THE MISSING WORD
'2'      'b' '2'   'A' '2'     '1'
```

GROUP FRAMES FOR THE SELECTED SENTENCE

1 1-2 2 3-4 3 4-7 4 8-8 5 9-10 6 11-12 7 13-14
8 15-17

COMPUTER PRODUCED RESPONSES TO THE QUESTION SENTENCE

ENTRY	CODE	SENTENCE
WORD	'1'	2

Example 3

INPUT TEXT

WE CAN TEACH SKETCHING THROUGH PROGRAMMED INSTRUCTION. A YOUNG ARGIST IS GIVEN A PICTURE WITH SOME OF THE LINES IN THE SKETCH OMITTED. THE STUDENT IS SUPPOSED TO DRAW THE OMITTED LINES.IN EACH OF THESE EXAMPLES OF PROGRAMMING, THE STUDENT MAKES SOME KIND OF AN OBSERVABLE RESPONSE. A CHARACTERISTIC OF PROGRAMED INSTRUCTION IS THAT THE STUDENT MAKES OBSERVABLE RESPONSES.YOU ARE RESPONDING WHEN YOU DO SOMETHING. WITH PROGRAMMED INSTRUCTIONS, YOU ARE REQUIRED TO MAKE OBSERVABLE RESPONSES.WHEN YOU LISTEN TO A LECTURE YOU PROBABLY RESPOND TO WHAT YOU HEAR, BUT SUCH RESPONSES CAN NOT BE OBSERVED. PROGRAMED INSTRUCTION, HOWEVER, REQUIRES THE STUDENT TO MAKE OBSERVABLE RESPONSES. PROGRAMMING SHARES THIS ADVANTAGE BY REQUIRING THE STUDENT TO MAKE OBSERVABLE ≠.

GRAMMATICAL ANALYSIS OF QUESTION SENTENCE

```
PROGRAMMING SHARES THIS ADVANTAGE BY  REQUIRING THE STUDENT
'2'         '1'    'A'  '1'       'F' '1'       'A' '1'
TO MAKE OBSERVABLE ≠
'B' '2' '3'        '1'
```

GROUP FRAMES OF QUESTION SENTENCE

1 1-4 2 5-6 3 7-8 4 9-10 5 11-11 6 12-12

TEST FRAME

WORD	CODE
SHARES	'2'
ADVANTAGE	'1'
REQUIRING	'1'
STUDENT	'1'
MAKE	'2'
OBSERVABLE	'3'

SELECTED SENTENCE AND ITS CODES

```
PROGRAMMED INSTRUCTION HOWEVER REQUIRES THE STUDENT  TO
'2'        '1'         'J'     '2'      'A' '1'      'B'
MAKE OBSERVABLE RESPONSES
'2'  '3'        '1'
```

GROUP FRAMES FOR THE SELECTED SENTENCE

1 1-2 2 3-3 3 4-4 4 5-6 5 7-8 6 9-9 7 10-10

COMPUTER PRODUCED RESPONSES FOR THE QUESTION SENTENCE

ENTRY	CODE	SENTENCE
INSTRUCTION	'1'	9
OBSERVABLE	'3'	0
RESPONSES	'1'	9

Example 4

INPUT TEXT

IMMEDIATELY TELLING THE STUDENT THAT A RESPONSE IS ACCEPTABLE INCREASES THE PROBABILITY THAT THE RESPONSE WILL OCCUR ON FUTURE OCCASIONS. THIS IS CALLED REINFORCING THE RESPONSE. WHEN WE TELL THE STUDENT IMMEDIATELY THAT A RESPONSE IS ACCEPTABLE WE REINFORCE THE RESPONSE. IMMEDIATE REINFORCEMENT OF ACCEPTABLE RESPONSES INCREASES THE PROBABILITY THAT THEY WILL OCCUR ON FUTURE OCCASIONS. TO INCREASE THE PROBABILITY OF AN ACCEPTABLE RESPONSE THE PROGRAMMER REINFORCES THE RESPONSES IMMEDIATELY AFTER IT HAS OCCURED. IMMEDIATE REINFORCEMENT OF ACCEPTABLE RESPONSES INCREASES THE PROBABILITY. TO INCREASE THE PROBABILITY OF AN ACCEPTABLE RESPONSE WE MUST REINFORCE IT IMMEDIATELY. IF WE ≠ REINFORCE A RESPONSE WE INCREASE THE PROBABILITY THAT IT WILL OCCUR ON FUTURE OCCASIONS.

GRAMMATICAL ANALYSIS OF THE QUESTION SENTENCE

```
IF  WE  ≠   REINFORCE A   RESPONSE WE  INCREASE THE PROBABILITY
'F' '1' '3' '1'       'A' '1'      '1' '2'      'A' '1'
THAT IT  WILL OCCUR ON  FUTURE OCCASIONS
'A'  '1' 'B'  '2'   'F' '3'    '1'
```

GROUP FRAMES FOR THE QUESTION SENTENCE

1 1-2 2 3-3 3 4-4 4 5-6 5 7-7 6 8-8 7 9-10
8 11-12 9 13-14 12 15-17

MUST FRAME

WORD	CODE
WE	'1'
REINFORCE	'1'
RESPONSE	'1'
WE	'1'
INCREASE	'2'
PROBABILITY	'1'
IT	'1'

SELECTED SENTENCE AND ITS CODES

TO	INCREASE	THE	PROBABILITY	OF	AN	ACCEPTABLE	RESPONSE
'1'	'2'	'A'	'1'	'F'	'A'	'3'	'1'

WE	MUST	REINFORCE	IT	IMMEDIATELY
'1'	'2'	'3'	'1'	'4'

GROUP FRAMES FOR THE SELECTED SENTENCE

1 1-2 2 3-4 3 5-8 4 9-9 5 10-10 6 11-11 7 12-12
8 13-13

COMPUTER PRODUCED RESPONSE FOR THE QUESTION SENTENCE

ENTRY	CODE	SENTENCE
ACCEPTABLE	'3'	7
MUST	'2'	7

APPENDIX B

SUFFIXES AND CODINGS

SUFFIX ENTRY	GRAMMAR CODE	ACTION CODES
SELVES	1	1
THING	1	1
WHERE	4	1
ANCE	1	1
MENT	1	61
SION	1	
NESS	1	61
BODY	1	1
SELF	1	1
LESS	3	1
LIKE	3	1
ABLE	3	1
IBLE	3	1
SOME	3	1
WARD	4	1
TIME	4	1
URE	1	21
ITY	1	21
AGE	1	1
ANT	1	21
ING	12	1
ISM	1	21
ONE	1	1
IZE	1	31
IFY	2	731

SUFFIX ENTRY	GRAMMAR CODE	ACTION CODES
ISH	3	41
OUS	3	5421
LAR	3	
ARY	3	1
FUL	3	31
ATE	3	521
ENT	3	421
ANT	3	421
IVE	3	21
ORY	3	521
EST	3	1
WAY	4	1
ILY	4	3
AL	13	621
SE	1	8
ER	13	421
TH	1	21
'S	3	1
S'	3	1
VE	2	
EN	23	1
ED	23	41
LY	34	1
IC	3	21
Y	12	421
S	12	1
D	12	1

APPENDIX C

DICTIONARY OF 1000 WORDS.

A	A AN	A ABLE	23 ABOUT	F
ARTICLE	1 ASIDE	14 ASK	2 AT	F
AFTERWARDS	4 AFTERNOON	13 AGAIN	F4 AGAINST	F
ABOVE	F43 ABSENT	23 ACCEPT	2 ACCIDENT	1
AS	43 ACCORD	12 ACCOUNT	12 ACCROSS	F4
ACT	12 ACTIVE	3 ACTION	1 ACTUAL	3
ADD	2 ADDRESS	12 ADJECTIVE	13 ADMIRE	2
ADVICE	1 AEROPLANE	1 AFRAID	3 AFTER	F
AGE	12 AGO	4 AGREE	2 AIM	2
AIR	12 ALL	A1 ALLOW	2 ALONE	3
ALONG	3 ALREADY	4 ALSO	4 ALWAYS	4
AMONG	4 AMOUNT	12 AMUSE	2 AND	EJ
ANGER	12 ANGLE	1 ANIMAL	1 ANOTHER	13
ANSWER	12 ANXIOUS	3 ANY	A1 APART	4
APPEAR	2 APPOINT	2 ARISE	2 ARM	12
AROUND	F ARRANGE	2 ARRIVE	2 ART	1
ATTACK	12 ATTEMPT	12 ATTEND	2 ATTENTION	1
AWAKE	8 AWAY	4 AWFULLY	D AXE	2
B	1 BACK	12 BAD	3 BAG	1
BETWEEN	F BEYOND	F BIG	3 BITE	2
BAND	12 BANK	12 BAR	1 BARREL	1
BATH	12 BATTLE	12 BE	B BEAM	1
BEAT	2 BEAUTY	1 BECAUSE	J BECOME	2
BEFORE	J BEG	2 BEGIN	2 BEHAVIOUR	1
BEHAVE	2 BEHIND	F BELIEF	1 BELIEVE	2
BELONG	2 BELOW	F BEST	3 BETTER	3
BLACK	13 BLADE	1 BLOCK	12 BLUE	13
BOARD	12 BODY	1 BOLD	3 BOOK	12
BOOT	1 BORDER	1 BOTH	A1 BOTTOM	1
BOWEL	1 BOX	1 BOY	1 BRAIN	1
BRANCH	12 BREADTH	13 BREAK	12 BRIGHT	3
BRING	2 BROAD	13 BROWN	3 BUILD	2
BUSH	12 BUSINESS	1 BUSY	23 BUTTER	1
BUT	E BUY	2 BY	F	
C	1 CALCULATE	2 CALL	12 CAN	2
COMPLAIN	2 COMPLETE	23 CONCERN	12 CONDITION	12
CONQUER	2 CONSIDER	2 CONTAIN	2 CONTINUE	2
CARRIAGE	1 CARRY	2 CASE	1 CATCH	2
CERTAIN	3 CHAIN	12 CHANCE	12 CHANGE	12
CAREFUL	3 CANDLE	1 CARD	1 CARE	2
CHARACTER	1 CHOICE	12 CHOOSE	2 CIRCLE	1
CONTROL	12 COPY	12 CORNER	1 CORRECT	23
CAUSE	1 CEASE	2 CENTER	1 CENTRE	1
CITY	1 CLAIM	2 CLASS	1 CLEAN	23
CLEAR	23 CLEVER	3 CLOCK	1 CLOSE	2
CLOUD	1 COLD	3 COLLECT	2 COLOUR	1
COME	1 COMMAND	12 COMMON	13 COMPASS	12
COULD	B COUNT	12 COURAGE	1 COURSE	1
COVER	12 CRACK	12 CROSS	12 CROWD	1
CUT	12 COST	12		

D	1	DAMAGE	12	DANGER	1	DARE	12
DETERMINE	2	DIFFERENCE	1	DIFFERENT	3	DIFFICULT	3
DECEIVE	2	DECIDE	2	DECLARE	2	DEED	1
DESERVE	2	DESIRE	12	DESTROY	2	DESTRUCTION	1
DISCOVER	2	DISTANCE	1	DIVIDE	2	DO	2
DIFFICULTY	14	DIG	2	DIRECT	12	DISAPPEAR	2
DELIGHT	12	DEMAND	12	DEPEND	2	DESCRIBE	2
DAY	1	DEAD	23	DEAL	12	DECAY	12
DEEP	13	DEPTH	1	DEFEND	2	DELAY	2
DOWN	14	DRIVE	12	DROP	12	DURING	F
DUTY	1	DUST	1	DOES	2		
E	1	EACH	13	EARLY	4	EARTH	1
ENQUIRE	2	ENTER	2	EQUAL	3	ESCAPE	2
EVERYONE	1	EXACT	23	EXAMINE	2	EXAMPLE	1
EXPECT	2	EXERCISE	12	EXCEPT	2	EXPERIENCE	1
EASY	3	EDGE	12	EFFECT	12	EITHER	FJ
ELECT	2	ELECTRIC	3	ELSE	43	EMPLOY	2
EMPTY	23	END	12	ENJOY	2	ENOUGH	34
EVEN	13	EVENT	1	EVER	4	EVERY	A1
EXPLAIN	2	EXPRESS	12	EYE	1		
F	1	FACE	12	FACT	1	FAIL	2
FAINT	23	FAIR	13	FALL	2	FAMILY	1
FANCY	12	FAR	4	FAST	43	FATE	1
FAVOUR	12	FAULT	12	FEAR	12	FEED	2
FEEL	2	FELLOW	1	FEW	3	FIELD	12
FIGHT	12	FIGURE	1	FILL	12	FIND	2
FINE	3	FIRST	13	FIT	12	FIX	12
FLAG	1	FLAT	13	FLOW	12	FLY	2
FOLD	12	FOLLOW	2	FOOT	1	FOR	F
FORBID	2	FORCE	12	FOREIGN	3	FORGET	2
FORM	12	FRAME	12	FORWARD	3	FREE	2
FREEZE	2	FRESH	13	FROM	F	FRONT	12
FULFIL	2	FULL	13	FURNISH	2	FURTHER	43
FUTURE	13						
G	1	GAIN	12	GAME	13	GATE	1
GATHER	12	GENERAL	13	GENEROUS	3	GENTLE	3
GET	2	GOT	2	GIFT	1	GIVE	2
GAVE	2	GLAD	23	GLASS	1	GO	2
GOOD	3	GRAND	3	GRAVE	2	GREAT	3
GREET	2	GROUND	1	GROUP	1	GROW	2
GUARD	12	GUESS	12	GUIDE	12		
H	1	HAIR	1	HALF	1	HALL	1
HAND	12	HANDLE	1	HANG	2	HAPPEN	2
HARD	13	HARM	12	HASTE	1	HAVE	2
HAD	B	HE	1	HEAR	2	HEAVY	3
HEIGHT	1	HELP	12	HER	A1	HERE	1
HIGH	13	HIS	A1	HIT	12	HOLD	12
HOPE	12	HOUR	1	HOW	I	HURRY	2
HOWEVER	J						
I	1	IDEA	1	IF	F	ILL	2

IN	F	INSIDE	F4	INTO	F	INCH
IMAGINE	2	IMPORTANT	3	IMPOSSIBLE	13	IMPROVE
INCREASE	2	INSTEAD	4	INSTRUMENT	1	INTEND
INTENTION	1	INVITE	2	IT	1	IS
J	1	JOIN	2	JOURNEY	1	JUDGE
JUST	43					
K	1	KEEN	13	KEEP	2	KEPT
KEY	12	KIND	12	KNOW	2	KNOWLEDGE
L	1	LACK	12	LAND	12	LANGUAGE
LARGE	3	LAST	13	LATE	13	LAW
LAY	2	LEAD	12	LEAF	1	LEARN
LEAST	13	LEAVE	12	LEFT	12	LEND
LENGTH	1	LESS	13	LESSON	1	LET
LETTER	1	LEVEL	12	LIFE	1	LIFT
LIGHT	13	LIKE	12	LIMB	1	LINE
LIP	12	LIST	12	LISTEN	2	LITTLE
LIVE	2	LOAD	12	LOAF	2	LOCK
LOOSE	13	LOSE	2	LOST	2	LOT
LODGE	1	LOG	12	LONG	23	LOOK
LOUD	3	LOVE	1	LOW	3	LOYAL
M	1	MACHINE	1	MAD	3	MAIN
MOVE	12	MUCH	A1	MUST	2	MY
MAKE	2	MALE	1	MAN	1	MANNER
MAY	B	MEAN	3	MEAL	1	MEASURE
MASS	12	MATTER	1	MATERIAL	1	MANY
MEET	12	MEMBER	1	MEMORY	1	MESSAGE
MAP	1	MARCH	2	MARK	2	MARRIAGE
MICROSCOPE	1	MIDDLE	3	MIGHT	B	MILL
MISS	12	MIST	12	MIX	12	MISTAKE
MIXURE	1	MOMENT	1	MONEY	1	MONTH
MOON	1	MORNING	13	MOST	A1	MOTOR
MORE	A1					
N	1	NAME	12	NARROW	12	NATION
NATIVE	1	NATURAL	13	NEAR	43	NECESSARY
NEED	12	NEIGHBOUR	1	NEITHER	43	NONE
NEVER	4	NOR	E	NET	12	NEW
NEXT	3	NO	AD	NOISE	1	NORTH
NOT	CE	NOTE	12	NOTICE	12	NOUN
NOW	4	NUMBER	12	NICE	3	
O	1	OBEY	2	OBJECT	12	OBTAIN
OF	F	OFF	43	OFFER	12	OFTEN
OLD	3	ON	F	ONCE	43	ONE
ONLY	43	OPEN	12	OPINION	1	OPPOSITE
OR	EJ	ORDER	12	OTHER	43	OUGHT
OUR	1	OUT	13	OVER	F	OWE
OWN	3					
P	1	PACK	12	PAGE	1	PAIN
PARTICIPLE	1	PARTY	1	PASS	12	PAST
PAIR	12	PAPER	1	PART	12	PARTICULAR
PRESENCE	1	PRESERVE	2	PRESS	12	PRETEND

PREVENT	2	PRINT	12	PROBABLE	13	PRODUCE	2
PRACTISE	12	PRAISE	2	PRECIOUS	3	PREPARE	2
PATH	1	PAUSE	12	PAY	12	PECULIAR	13
PEOPLE	1	PERFECT	13	PERFORM	2	PERHAPS	4
PERMIT	12	PERSON	1	PICTURE	1	PIECE	12
PIN	12	PLACE	12	PLAIN	13	PLAN	12
PLAY	12	PLEASANT	3	PLENTY	13	PLURAL	13
POINT	12	POLE	1	POLITE	3	POOR	3
POSSES	2	POSSIBLE	13	POST	12	POWER	12
PROPER	3	PROTECT	2	PROVE	2	PROOF	1
PROVIDE	2	PUBLIC	13	PULL	12	PURE	12
PURPOSE	1	PUT	2				
Q	1	QUALITY	1	QUANTITY	1	QUARTER	1
QUESTION	1	QUICK	13	QUIET	12	QUITE	4
R	1	RACE	12	RAISE	12	RANGE	12
RECOGNISE	2	REFUSE	2	REGARD	12	REGULAR	3
RELATION	1	REMAIN	2	REMEMBER	2	REPAIR	12
RANK	12	RAPID	13	RATHER	4	REACH	12
REAL	3	REASON	1	RECEIPT	1	RECEIVE	2
REPEAT	2	REPORT	12	RESPECT	12	REST	12
RESULT	1	RETURN	12	RICH	3	RIDE	12
RIGHT	13	RING	12	RISE	2	RISK	12
ROAD	1	ROPE	12	ROUGH	13	ROUND	23
RUBBER	1	RULE	12	RUIN	12	RUN	2
RUSH	12						
S	1	SAD	23	SAFE	12	SAME	13
SEAT	1	SECOND	13	SEE	2	SEED	12
SEEM	2	SEIZE	2	SELF	2	SEND	12
SENSE	12	SEPERATE	2	SET	12	SETTLE	2
SHIELD	12	SHORT	13	SHOULD	2	SHOW	12
SIDE	12	SIGHT	12	SIGN	12	SILLY	3
SIMPLE	13	SINCE	F4	SINGLE	13	SIT	2
SIZE	12	SKILL	12	SLIDE	12	SLOW	2
SMALL	13	SO	J	SOFT	12	SOLID	13
SOME	AD	SOON	43	SORT	12	SOUND	12
SPACE	12	SPEAK	2	SPEED	12	SPEND	12
SQUARE	12	STAND	12	STAR	1	START	12
STATE	12	STATION	1	STAY	12	STEADY	12
STEM	12	STEEP	12	STEP	12	STILL	43
STOP	12	STORE	12	STRAIGHT	12	STRANGE	2
STRONG	3	SUBJECT	13	SUCCEED	12	SUCH	43
SUDDEN	3	SUPPLY	12	SUPPORT	12	SURE	3
STRENGTH	1	STRETCH	12	STRIKE	12	STROKE	12
SEVERAL	13	SHAPE	12	SHE	1	SHEET	1
SATISFY	2	SAVE	2	SAY	2	SCENE	1
SCIENCE	1	SEA	1	SEASON	1	SEARCH	12
SURFACE	12	STUDENT	1				
T	1	TABLE	1	TAKE	12	TALK	12
THROUGH	F4	THUS	4	TILL	F	TIME	12
TOWARDS	43	TRAP	12	TROUBLE	12	TRUE	13

TALL	3	TEACH	2	TEMPER	12	TENDER	12
TENSE	13	TERRIBLE	12	THANK	12	THE	A
THEM	J	THEREFORE	J	THERE	1	THESE	A1
THEY	1	THEN	1	THING	1	THINK	2
THIS	A1	THOROUGH	13	THOSE	A1	THOUGH	4
TO	BF	TOGETHER	4	TOO	4	TOP	12
TRY	12	TURN	12	THAT	A1	THAN	F
U	1	UNDER	F	UNDERSTAND	2	UNION	1
UNITE	2	UNITY	1	UNIVERSITY	1	UP	4
US	1	USE	12	USUAL	13	UNTIL	F
V	1	VALUE	12	VARIOUS	3	VERB	2
VERY	34	VIEW	12	VISIT	12	VOICE	1
W	1	WAIT	2	WAKE	12	WALL	1
WANDER	2	WANT	2	WAR	1	WARM	23
WARN	2	WATCH	12	WAVE	12	WAY	12
WE	1	WEAK	13	WEAR	12	WEAPON	1
WEEK	1	WEIGH	2	WELCOME	12	WELL	12
WHAT	I	WHEN	IJ	WHERE	J	WHETHER	I
WHICH	I	WHILE	J	WHO	J	WHOM	I
WHOSE	I	WHY	I	WILL	B	WISE	2
WITH	F	WITHOUT	F	WORK	1	WORSE	2
WOULD	D	WRITE	2	WRONG	1		
Y	1	YARD	1	YES	4	YET	4
YIELD	12	YOU	1	YOUR	A	YOUTH	1

APPENDIX D

PROGRAM LISTING

```
$IBJOB
$IBFTC MAIN
C        PROGRAMMED TEXT PROCESSING USING GRAMMATICAL ANALYSIS
      COMMON/NABETS/ALBETS(26,2)/NDTNRY/LIM,DCTNRY(1500,6)
      COMMON/NWORD/ICOUNT,WORD(51,6)/NPOINT/INP/NTEST/TEST/NARRAY/ARRAY
     1(10,2),SEARCH(500,6)/NSUFIX/SUFIX1(52,2)
      COMMON/NGROUP/KNPP,GROUPL(25,2)
      INTEGER ALBETS,DCTNRY,WORD,SUFIX1,BLANK,DOLLAR,ARRAY,SEARCH,ONE,
     1TWO,THREE
      INTEGER A,B,F
      LOGICAL TEST
      LOGICAL TEKHEY
      DATA A, B,F/1HA,1HB,1HF/
      DATA BLANK,DOLLAR,IRECOR,JSNT,LPTR,LLK/1H ,3H$$$,4*0/
      DATA ONE,TWO,THREE,IMARK/1H1,1H2,1H3,1H  /
1000  FORMAT(I2/24(A1,I2))
 1001 FORMAT(4(2A6,A3,A3))
 1003 FORMAT(12A6)
1515  FORMAT(2X,21A6)
1517  FORMAT(*  THE   CAN BE FILLED WITH */*  ENTRY *,18X,*SENTENCE*/)
 1518 FORMAT(1X,4A6,I4)
 4000 FORMAT(50X,*FRAME WORDS*//20(50X,4A6/))
 6000 FORMAT(/20X,*SEARCH ARRAY IS OVERFLOWING*)
 6001 FORMAT(/20X,*CHECK ARRAY IS OVERFLOWING*)
 6002 FORMAT(20X,*FRAME ARRAY IS OVERFLOWING*)
 6003 FORMAT(/20X,*STORE ARRAY IS OVERFLOWING*)
 9999 FORMAT(*  ERROR IN PICKUP ROUTINE    ,GROUP FRAMES NOT PROPERLY PIC
     1KED*/)
      INP=65
C     INPUT DICTIONARY AND SUFFIX ENTRIES
      READ1003,((SUFIX1(I,J),J=1,2),I=1,52)
      READ1000,KLPNT,((ALBETS(I,J),J=1,2),I=1,26)
      DO 100 I=1,26
      LPOINT=ALBETS(I,2)
      ALBETS(I,2)=KLPNT
      KLPNT=KLPNT+4*LPOINT
  100 CONTINUE
      LIM=KLPNT-1
      DO 200 I=1,LIM,4
      I1=I+3
  200 READ1001,((DCTNRY(L,J),J=1,4),L=I,I1)
C     INITIALISATION OF THE ARRAY FOR STORING THE SENTENCE
1     DO 300 I=1,51
```

```
    DO 300 J=1,6
300 WORD(I,J)=BLANK
    ICOUNT=0
    TEXT READ IN (SENTENCE BEGINNING WITH  $$$ DENOTES THE END OF ALL
    INPUT TEXT
2   CALL INPUT
    IF(WORD(1,1).EQ.DOLLAR) GOTO 15
    IF(ICOUNT.EQ.0) GOTO 2
    JSNT=JSNT+1

          ARRAY(JSNT,1)=IRECOR+1
          DO 1400 I=1,ICOUNT
          IRECOR=IRECOR+1
          IF(IRECOR.GT.500) GOTO 5000
   C       CHECK FOR DIMENSION OVERFLOW OF 'SEARCH'
   C      INPUT TEXT STORED FOR LATER REFERENCE
          DO 1401 J=1,6
    1401  SEARCH(IRECOR,J)=WORD(I,J)
    1400  CONTINUE
          ARRAY(JSNT,2)=IRECOR
   C      CHECK IF INPUT TEXYT HAS MORE THAN 10 SENTENCES(INCLUDING QUESTION
   C      SENTENCE)
          IF(JSNT.EQ.10.AND.(.NOT.TEST))GO TO 1770
   C      CHECK FOR THE PRESENCE OF QUESTION SENTENCE
          IF(.NOT.TEST) GOTO 1
   C      GET THE GRAMMATICAL ANALYSIS OF THE QUESTION SENTENCE AND ITS
   C      GROUP FRAMES
          CALL AMBGTY(0)
          CALL PICKUP
   C      STORE THE GROUP FRAME OF THE   ENTRY IN CHECK FROM THE GROUPS OF T
   C      HE QUESTION SENTENCE
          DO 1500 I=1,ICOUNT
          IF(WORD(I,1).NE.IMARK) GOTO 1500
          DO 2000 J=1,KNPP
          IF(I.GT.GROUPL(J,2)) GOTO 2000
          GOTO 2001
    2000  CONTINUE
    2001  IT1=GROUPL(J,1)
          IT2=GROUPL(J,2)-1
          IGC=0
          DO 2002 K=IT1,IT2
          IGC=IGC+1
          IF(IGC.GT.10) GOTO 5001
   C      CHECK FOR THE DIMENSION OVERFLOW OF 'CHECK'
    2002  CHECK(IGC)=WORD(K,4)
   C      CHECK IF   IS THE LAST WORD IN THE QUESTION SENTENCE
          IF(I.EQ.ICOUNT) GOTO 2003
   C      STORE THE WORD THAT FOLLOWS    IN THE QUESTION SENTENCE
          BEGN(1)=WORD(I+1,1)
          BEGN(2)=WORD(I+1,2)
          BEGN(3)=WORD(I+1,3)
```

```
 2003         IC1=I-10
      IC2=I+10
      IF(IC1.LE.0) IC1=1
      IF(IC2.GT.ICOUNT) IC2=ICOUNT
C     STORE THE GRAMMAR CODE OF THE       IN 'LOC'
      LOC=WORD(I,4)
      GOTO 2500
 1500 CONTINUE
C     STORE THE TEST FRAME OF WORDS IN ' FRAME'(CHOOSE ONLY CLASS 1 OR
C     CLASS 2 OR CLASS 3 WORDS ONLY)
 2500 DO 1501 I=IC1,IC2
      IF(WORD(I,1).EQ.IMARK) GOTO 1501
      IF(WORD(I,4).EQ.ONE.OR.WORD(I,4).EQ.TWO.OR.WORD(I,4).EQ.THREE)
     1GOTO 1402
      GOTO 1501
 1402 LPTR=LPTR+1
      IF(LPTR.GT.20) GOTO 5002
C     CHECK FOTR THE DIMENSION OVERFLOW OF 'FRAME'
      DO 1503 J=1,4
 1503 FRAME(LPTR,J)=WORD(I,J)
 1501 CONTINUE
      PRINT4000,((FRAME(I,J),J=1,4),I=1,LPTR)
C     INITIALISE POINTERS FOR OUTPUT RESPONSES OF THE PRESENT TEXT
      LLK=0
      ISNT=JSNT-1
      DO 1111 I=1,JSNT
 1111 TABLE(I)=0
C     STORE THE NO. OF WORDS IN EACH SENTENCE THAT PROVIDE A MATCH TO
C     THE TEST FRAME OF WORDS IN 'TABLE'
      DO 1504 L=1,ISNT
      ICNT=0
      KK1=ARRAY(L,1)
      KK2=ARRAY(L,2)
      DO 1511 LK=1,LPTR
      DO 1505 I=KK1,KK2
      DO 1506 J=1,3
      IF(SEARCH(I,J).NE.FRAME(LK,J)) GOTO 1505
 1506 CONTINUE
      ICNT=ICNT+1
      TABLE(L)=ICNT
      GOTO 1511
 1505 CONTINUE
 1511 CONTINUE
 1504 CONTINUE
C     DETERMINE THE MAXIMUM WORD MATCH AMONG HHE SENTENCES
      MAX=TABLE(1)
      DO 1600 I=2,ISNT
      IF(TABLE(I).GT.MAX) MAX=TABLE(I)
 1600 CONTINUE
      DO 1602 I=1,ISNT
```

```
      IF(TABLE(I).NE.MAX) GOTO 1602
C     GET THE GRAMMATICAL ANALYSIS AND THE GROUP FRAMES OF THE SELECTED
C     SENTENCE
      CALL AMBGTY(I)
      CALL PICKUP
C     PASS 1 ANALYSIS TO DETERMINE THE DESIRED RESPONSE
      IHEY=0
      TEKHEY=.FALSE.
1906  DO 1507 K=1,ICOUNT
      IF(LOC.EQ.THREE)GO TO 1690
      IF(WORD(K,4).NE.LOC) GOTO 1507
      IF(LOC.EQ.ONE) GOTO 3000
C     ANALYSIS FOR THE GRAMMAR CODE OF CLASS 2 AND CLASS 4
      GO TO 1680
C     ANALYSIS FOR THE GRAMMAR CODE OF CLASS 3
1690  IF((IHEY.EQ.1).AND.(WORD(K,4).EQ.TWO.OR.WORD(K,4).EQ.THREE))
     1GO TO 1680
      IF(WORD(K,4).EQ.TWO.OR.WORD(K,4).EQ.THREE)GO TO 1900
      GO TO 1507
1900  DO 1901 LZZ=1,KNPP
      IF(K.GT.GROUPL(LZZ,2))GO TO 1901
      LKP=GROUPL(LZZ,2)
      IF(WORD(LKP,4).NE.ONE)GO TO 1507
      DO 1902  KLZ=1,3
      IF(BEGN(KLZ).NE.WORD(LKP,KLZ))GO TO 1507
1902  CONTINUE
C     ANALYSIS FOR CLASS 3 SUCCEEDS IN FIRST PASS
      TEKHEY=.TRUE.
      GO TO 1680
1901  CONTINUE
      PRINT 9999
      STOP
C     ANALYSIS FOR THE GRAMMAR CODE OF CLASS 1 FOR    ENTRY
 3000 IF(IHEY.EQ.1) GOTO 1680
      DO 3001 LZZ=1,KNPP
      IF(K.GT.GROUPL(LZZ,2)) GOTO 3001
      LKP1=GROUPL(LZZ,1)
      LKP2=GROUPL(LZZ,2)
      IF(WORD(LKP2,4).NE.ONE) GOTO 1507
      GOTO 3004
 3001 CONTINUE
      PRINT9999
      STOP
 3004 DO 3002    KLZ=1,IGC
      IF(CHECK(KLZ).EQ.A.OR.CHECK(KLZ).EQ.B.OR.CHECK(KLZ).EQ.F)GOTO 3002
      DO 3003 KLLZ=LKP1,LKP2
      IF(CHECK(KLZ).NE.WORD(KLLZ,4)) GOTO 3003
C     ANALYSIS FOR CLASS 1 SUCCEEDS   IN THE FIRST PASS
      TEKHEY=.TRUE.
      GOTO 1680
```

```
 3003 CONTINUE
 3002 CONTINUE
      GOTO 1507
C     CHECK IF SELECTED RESPONSE IS NOT ALREADY IN 'TEST FRAME'
 1680 DO 1500 LM=1,LPTR
      DO 1508 LK=1,4
      IF(WORD(K,LK).NE.FRAME(LM,LK)) GOTO 1500
 1508 CONTINUE
      GOTO 1507
 1500 CONTINUE
C     IF THE RESPONSE IS THE FIRST ONE STORE IT DIRECTLY
      IF(LLK.EQ.0) GOTO 1492
C     CHECK IF THE SELECTED RESPONSE HAS ALREADY BEEN TAKEN CARE OF
      DO 1531 III=1,LLK
      DO 1513 JJJ=1,3
      IF(WORD(K,JJJ).NE.STORE(III,JJJ)) GOTO 1531
 1513 CONTINUE
      GOTO 1507
 1531 CONTINUE
 1492 LLK=LLK+1
      IF(LLK.GT.200) GOTO 5003
C     CHECJ FOR THE DIMENSION OVERFLOW OF ' STORE'
      IF(WORD(K,4).EQ.ONE)GO TO 1700
      GO TO 1703
C     IF THE SELECTED RESPONSE IS CLASS 1 WORD,THEN COLLECT
C     ALL ITS QUALIFIERS(CLASS 3 WORDS) FROM THE SELECTED SENTENCE
1700  MTST=K
1701  MTST=MTST-1
      IF(WORD(MTST,4).NE.THREE)GO TO 1703
      DO 1702 LK=1,4
1702  STORE(LLK,LK)=WORD(MTST,LK)
      STORE(LLK,5)=0
      LLK=LLK+1
      IF(LLK.GT.200) GOTO 5003
      GO TO 1701
C        RE THE RESPONSE FOR PRINTING
 1703 DO 1510 LK=1,4
 1510 STORE(LLK,LK)=WORD(K,LK)
      STORE(LLK,5)=1
 1507 CONTINUE
      IF(IHEY.EQ.1)GO TO 1902
      IF(LOC.EQ.THREE.AND.(.NOT.TEKHEY))GO TO 1905
      IF(LOC.EQ.ONE.AND.(.NOT.TEKHEY)) GOTO 1905
C     GOTO ANALYSE OTHER TEXT SENTENCES THAT PROVIDE MAX WORD MATCH
      GOTO 1602
C     PASS 1 ANALYSIS TO FIX THE RESPONSE FAILS
C     DO PASS 2 ANALYSIS BY COLLECTING ALL WORDS OF THE SAME CATEGORY AS
C     THE   ENTRY OF THE QUESTION SENTENCE
 1905  IHEY=1
      GOTO 1900
```

```
 1602 CONTINUE
C      PRINT THE QUESTION SENTENCE AND THE SELECTED RESPONSES FOTR IT
       LK1=ARRAY(JSNT,1)
       LK2=ARRAY(JSNT,2)
       PRINT1515,((SEARCH(I,J),J=1,3),I=LK1,LK2)
       PRINT1517
       DO 1516 I=1,LLK
 1516  PRINT1518,(STORE(I,J),J=1,5)
C      INITIALISATION OF POINTERS AFTER THE PROVESSING OF A TEXT TO CONSI
C       DEROTHER TEXTS ALSO
 1770  JSNT=0
       IRECOR=0
       LPTR=0
       GOTO 1
C      ERROR MESSAGES TO DETECT DIMENSION OVERFLOW
 5000  PRINT6000
       GOTO 15
 5001  PRINT6001
       GOTO 15
 5002  PRINT6002
       GOTO 15
 5003  PRINT6003
  15   STOP
       END
$IBFTC PICKUP
       SUBROUTINE PICKUP
C      THIS ROUTINE PICKS THE GROUP FRAMES OF A SENTENCE ACCORDING TO
C      SPECIFIC GRAMMAR FRAMES
       COMMON/NGROUP/KNPP,GROUPL(25,2)
       COMMON /NWORD/ICOUNT,WORD(51,6)
       INTEGER GROUPL
       INTEGER H,G,II
       INTEGER WORD,GROUP(25)
       INTEGER A,B,F,ONE,TWO
       DATA A,B,F,ONE,TWO/1HA,1HB,1HF,1H1,1H2/
       DATA H,G,II/1HH,1HG,1HI/
 102    FORMAT(*  GROUP FRAMES OF THE SENTENCE*/*  GROUPS  WORDS  */)
 1000   FORMAT(*  ERROR IN GRAMMATICAL ANALYSIS *)
       KNP=0
       IC=1
C       SCAN THE SENTENCE TO SEPARATE IT INTO GROUPS ACCORDING TO SPECIFI
C      FRAMES
  101  DO 10 I=IC,ICOUNT
       IF(WORD(I,4).EQ.A)GO TO 11
       IF(WORD(I,4).EQ.B)GO TO 12
       IF(WORD(I,4).EQ.F)GO TO 11
       IF(I.EQ.1.AND.(.NOT.(WORD(I,4).EQ.H.OR.WORD(I,4).EQ.G.OR.WORD(I,4)
      1.EQ.II)))GO TO 11
C      GROUP OF WORDS NOT FALLING INTO ANY OF THE SPECIFIED GRAMMAR FRAME
       KNP=KNP+1
```

```
$IBFTC INPUT
C      LEXICON FOR THE PROGRAMMED TEXT INPUT
       SUBROUTINE INPUT
       DIMENSION TAB(15),LINE(80),LOCAL(18)
       COMMON/NTEST/TEST
       COMMON/NWORD/ICOUNT,WORD(51,6)/NPOINT/INP
       INTEGER HASHF
       INTEGER RPAREN,BLANK,SWITCH,COMMA,PERIOD,TAB,CHRCNT,WORD,STAR,CHK
       LOGICAL LOG
       LOGICAL TEST
       DATA MAX,LPAREN,RPAREN,BLANK,COMMA,PERIOD,STAR,NIL
      1/65,1H(,1H),1H ,1H,,1H.,1H*,1HN/
       DATA TAB/1H*,1H>,1H=,1H^,1H/,1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7
      1,1H8,1H9/
       DATA HASHF/1H /
1000   FORMAT(80X/80A1)
1001   FORMAT(3A6)
1002   FORMAT(80A1)
1003   FORMAT(1X,78A1)
1005   FORMAT(* LENGTH OF WORD ' *,18A1,* ' EXCEEDS 18 CHARACTERS*/
      1* LENGTH TRUNCATED TO 18 CHARACTERS AND PROCESSING PROCEEDS*)
1006   FORMAT(* NUMBER OF WORDS IN SENTENCE EXCEEDS 50.  EXECUTION
      1DELETED .*)
       SWITCH=1
       TEST=.FALSE.
       LOG=.TRUE.
C      SCAN THE INPUT CARD CHARACTERWISE
1      INP=INP+1
C      CHECK FOR END OF AN INPUT CARD
       IF(INP.GT.MAX)GO TO 13
       IF(LINE(INP).EQ.LPAREN.OR.LINE(INP).EQ.RPAREN)LINE(INP)=BLANK
       GO TO (2,3,4,5),SWITCH
C      PERIODS OR COMMA AT THE BEGINNIRG OF TEXT IGNORED
2      IF(LINE(INP).EQ.BLANK.OR.LINE(INP).EQ.COMMA)GO TO 1
       IF(LINE(INP).EQ.PERIOD)GO TO 1
21     SWITCH=2
C      CHECK IF INPUT CHARACTER IS AN ALPHABET OR SPECIAL CHARACTER
       DO 100 I=1,15
       IF(LINE(INP).EQ.TAB(I))SWITCH=3
100    CONTINUE
       GO TO 6
C      GETTING ALPHABETIC CHARACTERS,CHECK FOR END OF A WORD
3      IF(LINE(INP).EQ.BLANK)GO TO 7
       IF(LINE(INP).EQ.PERIOD.OR.LINE(INP).EQ.COMMA)GO TO 8
       GO TO 21
C      GETTING ALPHANUMERIC CHARACTERS,CHECK FOR END OF A WORD
4      IF(LINE(INP).EQ.BLANK)GO TO 7
       IF(LINE(INP).EQ.COMMA)GO TO 9
       IF(LINE(INP).EQ.PERIOD)GO TO 12
       GO TO 6
```

```
C  SKIP SUCCESSIVE BLANKS OR COMMAS AFTER A WORD
5      IF(LINE(INP).EQ.BLANK)GO TO 1
       IF(LINE(INP).EQ.COMMA.OR.LINE(INP).EQ.PERIOD)GO TO 81
       GO TO 21
C      CHECK FOR MORE THAN 18 CHARACTERS IN A WORD
6      IF(LOG)CHRCNT=CHRCNT+1
       IF(CHRCNT.GT.18)LOG=.FALSE.
       LOCAL(CHRCNT)=LINE(INP)
       GO TO 1
7      IF(.NOT.LOG)CHRCNT=CHRCNT-1
       IF(.NOT.LOG)PRINT 1005,(LOCAL(I),I=1,CHRCNT)
       IF(CHRCNT.EQ.0)GO TO 71
       LOG=.TRUE.
C      STORE THE WORD READ
       ICOUNT=ICOUNT+1
       IF(ICOUNT.GT.50)GO TO 98
       WRITE(99,1000)(LOCAL(I),I=1,CHRCNT)
       READ(99,1001)(WORD(ICOUNT,I),I=1,3)
       IF(SWITCH.EQ.3)WORD(ICOUNT,4)=3
C      CHECK FOR THE 0-2-8 PUNCH(SPECIAL MARKER FOR THE BLANK ENTRY
C      SENTENCE READ)
       IF(WORD(ICOUNT,1).EQ.HASHE)TEST=.TRUE.
       IF(WORD(ICOUNT,1).EQ.HASHE)WORD(ICOUNT,4)=3
C      DO DICTIONARY LOOKUP FOR THE PRESENT WORD READ
       IF(WORD(ICOUNT,4).EQ.BLANK)CALL LOOKUP(ICOUNT)
       CHRCNT=0
71     IF(LINE(INP).EQ.COMMA)GO TO 9
C      LOOKING FOR END OF A SENTENCE
       IF(LINE(INP).EQ.PERIOD)GO TO 11
       SWITCH=4
       GO TO 1
8      WORD(ICOUNT+2,1)=LINE(INP)
       GO TO 7
81     WORD(ICOUNT+1,1)=LINE(INP)
       GO TO 71
  9    SWITCH=4
       GO TO 1
10     WORD(ICOUNT+1,1)=LINE(INP)
       GO TO 9
11     RETURN
12     IF(LINE(INP+1).EQ.BLANK)GO TO    8
       GO TO 6
C      READ A FRESH CARD AND CHECK FOR PRESENCE OF COMMENT CARDS(A * IN C
13     READ 1002,CHK,(LINE(I),I=1,MAX)
       PRINT 1003,(LINE(I),I=1,MAX)
       IF(STAR.EQ.CHK)GO TO 13
       INP=0
       GO TO 1
98     PRINT 1006
       STOP
```

```
      END
$IBFTC OUTPUT
C     THIS ROUTINE PRINTS THE CONTENTS OF THE WORD ARRAY SIX PER LINE
      SUBROUTINE OUTPUT
      COMMON/NWORD/ICOUNT,WORD(51,6)
      INTEGER WORD
1000  FORMAT(//1X,21A6)
1001  FORMAT(1X,7(A6,12X))
      DO 100 I=1,ICOUNT,7
      KK=I+6
      IF(KK.GT.ICOUNT)KK=ICOUNT
      PRINT 1000,((WORD(K,J),J=1,3),K=I,KK)
100   PRINT 1001,(WORD(K,4),K=I,KK)
      RETURN
      END
$IBFTC LOOKUP
C     DICTIONARY LOOKUP FOR THE GIVEN       WORD(ARGUMENT)
      SUBROUTINE LOOKUP(N)
      COMMON/NWORD/ICOUNT,WORD(51,6)/NROOT/ROOT(3)/NABETS/ALBETS(26,2)/N
     1DTNRY/LIM,DCTNRY(1500,4)/NFOUND/FOUND
      COMMON/NVALUE/IVALUE
      INTEGER WORD,ROOT,TEMP,ALBETS,DCTNRY
      LOGICAL FOUND
1000  FORMAT(A6)
1001  FORMAT(A1)
      IVALUE=0
      IF(N.EQ.0)GO TO 6
C     LOOKUP CALLED FROM INPUT TO FIND A MATCH TO GIVEN WORD ITSELF
      DO 300 I=1,3
300   ROOT(I)=WORD(N,I)
C     LOOKUP CALLED FROM SUFIX ROUTINE TO FIND A MATCH TO ROOT OF THE
C     GIVEN WORD
6     WRITE(99,1000)ROOT(1)
      READ(99,1001)TEMP
C     DETERMINE THE LIMITS OF THE DICTIONARY SCAN
      LIM1=1
      LIM2=LIM
      DO 100 I=1,26
      IF(ALBETS(I,1).EQ.TEMP)GO TO 1
100   CONTINUE
      LIM2=ALBETS(1,2)-1
      GO TO 2
1     LIM1=ALBETS(I,2)
      IF(I.EQ.26)GO TO 2
      LIM2=ALBETS(I+1,2)-1
C     SEARCH THE DICTIONARY IN THE DETERMINED LIMITS
2      DO 200 I=LIM1,LIM2
      IF(ROOT(1).NE.DCTNRY(I,1))GO TO 200
      IF(ROOT(2).NE.DCTNRY(I,2))GO TO 200
      IF(ROOT(3).EQ.DCTNRY(I,3))GO TO 3
```

```
200   CONTINUE
C     DICTIONARY SEARCH FAILED
      FOUND=.FALSE.
      RETURN
C     SEARCH SUCCEEDS,SAVE GRAMMAR OF DICTIONARY ENTRYY
3     FOUND=.TRUE.
      IF(N.EQ.0)IVALUE=DCTNRY(I,4)
      IF(N.EQ.0)RETURN
4     WORD(N,4)=DCTNRY(I,4)
      RETURN
      END
$IBFTC STRIP
C     THIS ROUTINE STRIPS N CHARACTERS FROM WORD(M) AND SAVES IT IN ENDI
      SUBROUTINE STRIP(M,N)
      DIMENSION WORD(51,6),ROOT(3),TEMP(18)
      COMMON/NWORD/ICOUNT,WORD/NROOT/ROOT/NEDING/ENDING/NTEMP/TEMP
      INTEGER WORD,ROOT,ENDING,TEMP,BLANK
      DATA BLANK/1H /
1000  FORMAT(3A6)
1001  FORMAT(18A1)
1002  FORMAT(18X/18A1)
      WRITE(99,1000)(WORD(M,K),K=1,3)
      READ(99,1001)(TEMP(K),K=1,18)
      K1=0
      DO 100 I=1,18
      IF(TEMP(I).EQ.BLANK)GO TO 1
      K1=K1+1
100   CONTINUE
1     K2=K1-N
      IF(K2.LE.0) GOTO 2
C     REMOVED CHARACTERS STORED IN ENDING
      K3=K2+1
      WRITE(99,1002)(TEMP(I),I=1,K2)
      READ(99,1000)(ROOT(I),I=1,3)
      WRITE(99,1002)(TEMP(I),I=K3,K1)
      READ(99,1000)ENDING
      RETURN
C     STRIPPING N CHARACTERS NOT POSSIBLE
  2   ROOT(1)=WORD(M,1)
      ROOT(2)=WORD(M,2)
      ROOT(3)=WORD(M,3)
      ENDING=BLANK
      RETURN
      END
$IBFTC ADD
C     THIS ROUTINE DELETES M CHARACTERS AND ADDS A WORD CHAR OF N
C     CHARACTERS FOR THE ROOT
      SUBROUTINE ADD(M,N,CHAR)
      DIMENSION TEMP(18),SDR(6)
      COMMON/NROOT/ROOT(3)
```

```
      COMMON/NK2/K2/NTEMP/TEMP
      INTEGER ROOT
      INTEGER SOR,TEMP,CHAR
1000  FORMAT(A6)
1001  FORMAT(6A1)
1002  FORMAT(3A6)
1003  FORMAT(18A1)
1004  FORMAT(18X/18A1)
      WRITE(99,1002)(ROOT(I),I=1,3)
      READ(99,1003)(TEMP(I),I=1,18)
C     K2   SPECIFIES NUMBER OF CHARACTERS IN ROOT
      N2=K2
C     DELETION OF M CHARACTERS FROM ROOT
      N2=N2-M
      IF(N.EQ.0)GO TO 1
C     CHAR BEING ADDED TO ROOT
      WRITE(99,1000)CHAR
      READ(99,1001)(SOR(I),I=1,N)
      DO 100 I=1,N
      N2=N2+1
100   TEMP(N2)=SOR(I)
1     WRITE(99,1004)(TEMP(I),I=1,N2)
      READ(99,1002)(ROOT(I),I=1,3)
      RETURN
      END
$IBFTC REPLAE
C     THIS ROUTINE SAVES CONTENTS OF FIRST ARGYMENT IN SECOND ARGUMENT
      SUBROUTINE REPLAE(RIGHT,ALEFT)
      INTEGER RIGHT(3),ALEFT(3)
      DO 100 I=1,3
100   ALEFT(I)=RIGHT(I)
      RETURN
      END
$IBFTC SUFFIX
C     ROUTINE FOR SUFFIX ANALYSIS  OF WORD(N)
      SUBROUTINE SUFIX(N)
      COMMON/NWORD/ICOUNT,WORD(51,6)/NROOT/ROOT(3)/NENDING/ENDING/
     1NK2/K2/NTEMP/TEMP/NSUFIX/SUFIX1
      COMMON /NFOUND/FOUND
      INTEGER TEMP1(4)
      INTEGER TAB(7)
      INTEGER WORD,ROOT,ENDING,BLANK,TEMP(18),IEMP(9),ITEMP(3)
     1,SUFIX1(52,2),ITAB(8)
      LOGICAL FOUND
      DATA TAB/53,50,30,17,4,2,1/
      DATA ITAB/1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8/
      DATA IEMP/1HE,1HY,3HITY,1HD,3HIDE,2HOF,1HL,2HLE,2HLA/
      DATA BLANK/1H /
1000  FORMAT(3A6)
1001  FORMAT(18A1)
```

```
 1002 FORMAT(18X/18A1)
1003  FORMAT(2X,4A1)
1004  FORMAT(A2)
1005  FORMAT(*  ERROR IN ACTION CODE OF THE SUFFIX *,2X,A6)
      WRITE(99,1000)(WORD(N,I),I=1,3)
      READ(99,1001)(TEMP(I),I=1,18)
      NUM=7
C     SUFFIXES PICKED IN GROUPS(NUM) FOR ANALYSIS
1     NUM=NUM-1
      IF(NUM.EQ.0)RETURN
C     DETERMINE THE LIMITS OF PRESENT GROUP OF SUFFIXES
      LIM2=TAB(NUM)-1
      LIM1=TAB(NUM+1)
      K2=NUM
C     SEPARATE THE SUFFIX AND ROOT OF THE GIVEN WORD
      DO 100 I=1,18
      K1=19-I
      IF(TEMP(K1).EQ.BLANK)GO TO 100
      IF(NUM.EQ.K2)K3=K1
      K2=K2-1
      IF(K2.EQ.0)GO TO 2
100   CONTINUE
      GO TO 1
2     K2=K1-1
      IF(K2.LE.1)GO TO 1
      WRITE(99,1002)(TEMP(I),I=1,K2)
      READ(99,1000)(ROOT(I),I=1,3)
      WRITE(99,1002)(TEMP(I),I=K1,K3)
      READ(99,1000)ENDING
C     SAVE THE ROOT OF THE WORD
      CALL REPLAE(ROOT,ITEMP)
C     SEARCH THE SUFFIXES WITHIN THE LIMITS
      DO 200 I=LIM1,LIM2


      IF(ENDING.EQ.SUFIX1(I,1))GO TO 4
200   CONTINUE
C     GROUP SELECTED NOT HAVING A MATVH TO THE SUFFIX OF THE WORD
      GO TO 1
4     WRITE(99,1000)SUFIX1(I,2)
C     GROUO HAS A MATCH,SAVE ITS GRAMMAR CODE AND ACTION CODES
      READ(99,1004)WORD(N,4)
      READ(99,1003)(TEMP1(J),J=1,4)
C     CHECK IF SIFFIX MATCHED IS ONE OF ' SION,LAR,VE'
      IF(I.EQ.6)GO TO 14
      IF(I.EQ.28)GO TO 15
      IF(I.EQ.45)GO TO 16
C     SUFFIX IS NOT ONE OF SPECIAL CATEGORY
      DO 300 J=1,4
C     CHECK FOR ACT+ON CODES OF THE SUFFIX
      IF(TEMP1(J).EQ.BLANK)GO TO 300
      DO 400 L=1,8
```

```
C     BRANCH TO ONE OF THE ACTION CODES DETERMINED
      IF(TEMP1(J).EQ.ITAB(L))GO TO(5,6,7,8,9,10,11,12),L
400   CONTINUE
      GO TO 20
C     ACTION  CODE '1'
5     CALL LOOKUP(0)
      IF(FOUND)GO TO  17
      GO TO 13
C     ACT+ON  CODE '2'
6     CALL ADD(0,1,IEMP(1))
      GO TO 5
C     ACT+ON  CODE '3'
7     CALL ADD(0,1,IEMP(2))
      GO TO 5
C     ACT+ON  CODE     4
8     CALL ADD(1,0,0)
      GO TO 5
C     ACTION CODE 5
9     CALL ADD(1,1,IEMP(1))
      GO TO 5
C     ACTION CODE 6
10    CALL ADD(1,1,IEMP(2))
      GO TO 5
C     ACT+ON CODE 7
11    CALL ADD(0,3,IEMP(3))
      GO TO 5
C     ACT+ON CODE 8
12    CALL ADD(0,1,IEMP(4))
      GO TO 5
13    CALL REPLAE(ITEMP,ROOT)
300   CONTINUE
      GO TO 18
C     SUFFIX  'SION'    PROCESSED
14    CALL ADD(0,3,IEMP(5))
      CALL LOOKUP(0)
      IF(FOUND)GO TO 17
      CALL REPLAE(ITEMP,ROOT)
      CALL ADD(1,3,IEMP(5))
      GO TO 19
C     SUFFIX LAR PROCESSED
15    CALL ADD(0,1,IEMP(7))
      CALL LOOKUP(0)
      IF(FOUND)GO TO 17
      CALL REPLAE(ITEMP,ROOT)
      CALL ADD(0,2,IEMP(8))
      CALL LOOKUP(0)
      IF(FOUND)GO TO 17
      CALL REPLAE(ITEMP,ROOT)
      CALL ADD(0,2,IEMP(9))
      GO TO 19
```

```
C      SUFFIX  VE PROCESSED
16     CALL ADD(0,2,IEMP(6))
       GO TO 19
       COMMON/NVALUE/IVALUE
17     IF(IVALUE.EQ.0)RETURN
       IF(WORD(N,4).EQ.IVALUE)RETURN
C      CHOOSE THE INTERSECTION OF SUFFIX CODE AND DICTIONARY CODE
C      AS THE GRAMMAR CODE OF THE GIVEN WORD
       WRITE(99,1000)WORD(N,4)
       READ(99,1001)J11,J12
       IF(J12.EQ.BLANK)RETURN
       IF(J11.EQ.IVALUE.OR.J12.EQ.IVALUE)WORD(N,4)=IVALUE
18     RETURN
19     CALL LOOKUP(9)
       IF(FOUND)GO TO 17
       GO TO 18
C      ACTION CODES NOT ONE OF THE 8 CODES FIXED
   20 PRINT 1005,ENDING
       STOP
       END
$IBFTC AMBGTY
C      THIS ROUTINE REMOVES GRAMMATICAL CODE AMBIGUITIES
       SUBROUTINE AMBGTY(NNN)
       COMMON/NEDING/ENDING
       COMMON/NWORD/ICOUNT,WORD(51,6)/NABETS/ALBETS(26,2)
       COMMON/NARRAY/ARRAY(10,2),SEARCH(500,6)
       INTEGER S,ING,ENDING
       INTEGER WORD,ALBETS,BLANK,THERE,H,DID,G,TAB(7),BEB,NIL,AONE,ONE3,
      1THREE1,A,ONE,TWO,THREE
       INTEGER FOUR
       INTEGER BF,B   ,F
       INTEGER ONE2,TWO1
       INTEGER THREE4
       INTEGER ARRAY,SEARCH
       INTEGER THAT
       INTEGER EJ,E
       INTEGER TWO3
       DATA S,ING/1HS,3HING/
       DATA TAB/4HWHEN,5HWHERE,3HWHY,3HHOW,3HWHO,5HWHICH,4HWHAT/
       DATA FOUR/1H4/
       DATA BLANK,THERE,H,DID,G,NIL,AONE,ONE3,THREE1,A,ONE,TWO,THREE/1H ,
      15HTHERE,1HH,3HDID,1HG,1HN,2HA1,2H13,2H31,1HA,1H1,1H2,1H3/
       DATA BF,B/2HBF,1HB/
       DATA F/1HF/
       DATA ONE2,TWO1/2H12,2H21/
       DATA II/1HI/
       DATA THREE4/2H34/
       DATA THAT/4HTHAT/
       DATA EJ,E/2HEJ,1HE/
       DATA TWO3/2H23/
```

```
1003   FORMAT(12A6)
1004   FORMAT(6A1)
 2000  FORMAT(28X,*AFTER DICTIONARY LOOKUP ALONE*/)
 2010  FORMAT(24X,*  AFTER SUFIX TEST ALONE*/)
2020   FORMAT(25X,* AFTER TWO PASSES */)
C      CHECK IF ARRAY WORD  CONTAINS SENTENCE FOR WHICH AMBIGUITY IS TO B
C       MADE
       IF(NNN.EQ.0) GOTO 10
       ICOUNT=0
       KK1=ARRAY(NNN,1)
       KK2=ARRAY(NNN,2)
       DO 11 I=KK1,KK2
       ICOUNT=ICOUNT+1
       DO 11 J=1,4
   11   WORD(ICOUNT,J)=SEARCH(I,J)
C      PRINT AFTER DICTIONARY LOOKUP
  10   PRINT 2000
       IF(NNN.EQ.0) CALL OUTPUT
       DO 400 I=1,ICOUNT
       IF(WORD(I,4).NE.BLANK) GOTO 401
       CALL SUFIX(I)
       GOTO 400
  401  IF(WORD(I,4).EQ.3) WORD(I,4)=BLANK
  400  CONTINUE
C      PRINT AFTER SUFFIX TEST ALONE
       PRINT 2010
       IF(NNN.EQ.0)CALL OUTPUT
C      FIND IF FIRST WORD IS BELONGING TO SPECIAL GROUP
       IF(WORD(1,1).EQ.THERE)WORD(1,4)=H
       IF(WORD(1,1).EQ.DID) WORD(1,4)=G
       DO 500 I=1,7
       IF(WORD(1,1).EQ.TAB(I)) WORD(I,4)=II
  500  CONTINUE
C      RESOLVING CERTAIN SPECIAL COMBINED GROUPS LIKE 'BE'  '2'  'EJ'
C      AND 'A1'
       DO 800 I=1,ICOUNT
       IF(WORD(I,4).EQ.BE)GO TO 810
       IF(WORD(I,4).EQ.TWO)GO TO 813
       IF(WORD(I,4).EQ.EJ)WORD(I,4)=E
        IF(AONE.EQ.WORD(I,4))GO TO 131
       GO TO 800
810    WORD(I,4)=F
       IF(WORD(I-1,4).EQ.B)GO TO 812
       IF(WORD(I-1,4).EQ.TWO)GO TO 811
       IF(WORD(I+1,4).EQ.TWO.OR.WORD(I+1,4).EQ.B)GO TO 812
       IF(WORD(I+1,4).EQ.ONE2) GOTO 831
       GO TO 800
  831  WORD(I+1,4)=TWO
       GOTO 812
813    IF(WORD(I+1,4).EQ.ONE2.OR.WORD(I+1,4).EQ.TWO1)GO TO 812
```

```
      IF(WORD(I+1,4).EQ.TWO3)GO TO 812
      GO TO 800
811   WORD(I-1,4)=B
812   WORD(I,4)=B
      CALL STRIP(I,3)
      IF(ENDING.EQ.ING)GO TO 814
      GO TO 800
131   WORD(I,4)=ONE
      IF(WORD(I+1,4).EQ.ONE3.OR.WORD(I+1,4).EQ.ONE2.OR.WORD(I+1,4)
     1.EQ.BLANK.OR.WORD(I+1,4).EQ.THREE.OR.WORD(I+1,4).EQ.ONE)GO TO 140
      IF(WORD(I+1,4).EQ.A.AND.WORD(I,1).EQ.THAT)WORD(I,4)=ALBETS(10,1)
      GOTO 800
140   WORD(I,4)=A
      GOTO 800
814   CALL STRIP(I+1,1)
      IF(ENDING.EQ.S)WORD(I,4)=TWO
800   CONTINUE
C    ONE PASS AMBGUITY REMOVED
      REP=1
      IF(WORD(1,4).EQ.TWO.OR.WORD(1,4).EQ.THREE)GO TO 1315
      IF(WORD(1,4).EQ.ONE2.OR.WORD(1,4).EQ.TWO3)GO TO 1315
      IF(WORD(1,4).EQ.BLANK)GO TO 1315
      GO TO 3
C    FIRST WORD BELONGS TO GROUP OF WORDS FORMING FORM CLASS 1
1315  LLJ=0
      JJK=1
      CALL ENTRY(LLJ,JJK)
3     DO 600 I=1,ICOUNT
      IF(WORD(I,4).EQ.BLANK.OR.WORD(I,4).EQ.NIL)GO TO 600
      IF(WORD(I,4) .EQ.ONE.OR.WORD(I,4).EQ.TWO)GO TO 600
      IF(WORD(I,4).EQ.THREE.OR.WORD(I,4).EQ.FOUR)GO TO 600
31    DO 700 J=1,9
      IF(WORD(I,4).EQ.ALBETS(J,1))GO TO 4
700   CONTINUE
      GO TO 13
C   FUNCTIONAL UNIT GO FOR ANALYSIS
4     CALL ENTRY(I,J)
      GO TO 600
C    GROUP '13' IDENTIFIED FOR INTERPOLATION
13    IF(ONE3.EQ.WORD(I,4).OR.THREE1.EQ.WORD(I,4))GO TO 132
      GO TO 600
132   IF(I.EQ.ICOUNT)GO TO 137
      IF(WORD(I+1,4).EQ.BLANK)GO TO 136
      IF(WORD(I+1,4).EQ.ONE)GO TO 135
      IF(WORD(I+1,4).EQ.ONE3.OR.WORD(I+1,4).EQ.THREE1)GO TO 600
      WRITE(99,1003)WORD(I+1,4)
      READ(99,1004)IP11,IP22
      IF(IP11.EQ.THREE.OR.IP22.EQ.THREE)GO TO 138
      IF(IP11.EQ.ONE.OR.IP22.EQ.ONE)GO TO 139
137   WORD(I,4)=ONE
```

```
135    K112=I
       IF(I.EQ.1)GO TO 600
       IF(WORD(I,4).EQ.ONE3.OR.WORD(I,4).EQ.THREE1)WORD(I,4)=THREE
133    IF(WORD(K112-1,4).EQ.ONE3.OR.WORD(K112-1,4).EQ.THREE1)GO TO 134
       GO TO 600
138    WORD(I+1,4)=THREE1
       GO TO 600
139    WORD(I+1,4)=ONE
       GO TO 135
134    K112=K112-1
       WORD(K112,4)=THREE
       GO TO 133
136    WORD(I,4)=ONE3
600    CONTINUE
C    PASS TWO OVER OR NOT
       IF(REP.EQ.2)GO TO 14
C    FILL BLANK ENTRIES WITH GROUP '13'
       DO 610 I=1,ICOUNT
       IF(WORD(I,4).NE.BLANK)GO TO 610
       WORD(I,4)=ONE3
610    CONTINUE
       REP=2
       GO TO 3
C    PASS TWO OVER MAKE SOME DOUBLE GROUPS INTO SINGLE
14     DO 900 I=1,ICOUNT
       JJK=1
       IF(WORD(I,4).EQ.THREE)CALL ENTRY(I-1,JJK)
       IF(WORD(I,4).EQ.ONE2.OR.WORD(I,4).EQ.TWO3)WORD(I,4)=TWO
       IF(WORD(I,4).EQ.THREE4)WORD(I,4)=FOUR
900    CONTINUE
C     PRINT FINAL GRAMMER
       PRINT 2020
       IF(NNN.EQ.0)CALL OUTPUT
       RETURN
       END
$IBFTC ENTRY
C    FUNCTIONAL GROUP ANALYSIS DONE HERE
C    IN THIS ROUTINE
       SUBROUTINE ENTRY(I,MJM)
       COMMON/NWORD/ICOUNT,WORD(51,6)/LPNT/IPNT,TCNT/NEDING/ENDING
       INTEGER F
       INTEGER WORD,ENDING,TCNT(200,2),BLANK,ONE,TWO,THREE,ED
       INTEGER TAB(4),B
       INTEGER D,FOUR,THREE4
       INTEGER MSK2,A,ONEA
       INTEGER ONE3
       INTEGER ANOT,C,MSK3
       INTEGER G
       INTEGER S,ARE,WERE,THESE,THOSE,THE,JOY,ONE2
       DATA F/1HF/
```

```
      DATA TAD,B/3HHAD,3HGET,4HHAVE,4HKEEP,1HB/
      DATA ONE,TWO,THREE,MSK1,ING/1H1,1H2,1H3,0007760605060,3HING/
      DATA MSK4/0006060606060/
      DATA ED,BLANK/2HED,1H /
      DATA D,FOUR,THREE4/1HD,1H4,2H34/
      DATA MSK2,A,ONEA/0776060606060,1HA,2H1A/
      DATA ONE3/2H13/
      DATA ANOT,C,MSK3/3HNOT,1HC,2HA1/
      DATA G/1HG/
      DATA S,ARE,HERE,THESE,THOSE,THE/1HS,3HARE,4HHERE,5HTHESE,5HTHOSE
     1   ,3HTHE/,JOY,ONE2/1HJ,2H12/
1000  FORMAT(A6)
1001  FORMAT(6A1)
C   SWITCH TO THE APPROPRIATE FUNCTIONAL GROUP
      GO TO(101,201,301,401,501,601,701,801,901),MJM
C   FUNCTIONAL GROUP 'A' ANALYSIS
101   K1=1
11    K1=K1+1
      IF(WORD(K1,4).EQ.BLANK)GO TO 12
      IF(WORD(K1,4).EQ.ONE)GO TO 14
      IF(WORD(K1,4).EQ.TWO)GO TO 15
      IF(WORD(K1,4).EQ.THREE)GO TO 12
C   FIND IF LOCAL POINTER EXCEEDS INPUT LENGTH
      IF(K1.EQ.ICOUNT)GO TO 17
C   IS IT A SINGLE GRAMMER WORD
      K2=INTGER(AND(WORD(K1,4),MSK1))
      IF(K2.EQ.MSK4)GO TO 13
C   SEPERATE THE TWO GRAMMER PARTS
      WRITE(99,1000)WORD(K1,4)
      READ(99,1001)K3,K4
      IF(WORD(K1,4).EQ.ONE2.AND.WORD(K1+1,4).EQ.ONE2)GO TO 1080
      IF(K3.EQ.THREE.AND.K4.EQ.ONE)GO TO 12
      IF(K3.EQ.ONE.AND.K4.EQ.THREE)GO TO 12
      IF(K3.EQ.TWO.OR.K4.EQ.TWO)GO TO 20
103   IF(K3.EQ.THREE.OR.K4.EQ.THREE)GO TO 18
      IF(K3.EQ.ONE.OR.K4.EQ.ONE)GO TO 19
      GO TO 13
C   TWO CONSECUTIVE '12' GROUPS OCCUR
1080  CALL STRIP(K1+1,1)
      IF(ENDING.NE.S)GO TO 20
      CALL STRIP(K1,3)
      IF(ENDING.EQ.ING)GO TO 20
      CALL STRIP(K1,2)
      IF(ENDING.EQ.ED)GO TO 20
C   SINCE NON SATISFIED SET CURRENT WORD GROUP '1'
C   AND THE NEXT WORD GROUP '2' AND QUIT
      WORD(K1,4)=ONE
      WORD(K1+1,4)=TWO
      GO TO 14
C   SET GRAMMER TO '3'
```

```
18    WORD(K1,4)=THREE
      GO TO 12
C   SET GRAMMER TO '1' AND QUIT
19    WORD(K1,4)=ONE
      GO TO 14
C   GROUP '2' HAS OCCURED TEST FOR ENDING 'ING' OR 'ED' QUALIFIERS
20    CALL STRIP(K1,3)
      IF(ENDING.EQ.ING)GO TO 102
      CALL STRIP(K1,2)
      IF(ENDING.EQ.ED)GO TO 102
      CALL STRIP(K1,1)
      IF(ENDING.EQ.S)GO TO 104
      GO TO 103
C   IF ENDING IS 'S' FIND IF IT IS CLASS '1' OR '2'
104   IF(WORD(K1+1,4).EQ.BLANK)GO TO 105
      IF(WORD(K1+1,4).EQ.TWO.OR.WORD(K1+1,4).EQ.D.OR.WORD(K1+1,4).EQ.F
     1.OR.WORD(K1+1,4).EQ.JOY)GO TO 107
      IF(WORD(K1-1,4).EQ.A.OR.WORD(K1-1,4).EQ.THREE.OR.WORD(K1-1,4)
     1.EQ.TWO)GO TO 107
      GO TO 108
105   IF(WORD(I,1).EQ.THE.OR.WORD(I,1).EQ.THOSE.OR.WORD(I,1).EQ.THESE
     1)GO TO 106
C   SET GRAMMER TO '2'
108   WORD(K1,4)=TWO
      GO TO 13
C   SET CURRENT WORD CLASS TO '1' AND NEXT WORD TO '2'
106   WORD(K1+1,4)=TWO
107   WORD(K1,4)=ONE
      GO TO 14
C   SET CURRENT WORD TO CLASS '2' BELONGS TO ATTRIBUTE LIST
102   WORD(K1,4)=TWO
      GO TO 12
C   IF LOCAL POINTER MORE THAN THREE OR EXCEEDS ICOUNT
C   MAKE UP PROPER SETTINGS AND QUIT
12    IF(K1.GT.(I+3))GO TO 13
      IF(K1.EQ.ICOUNT)GO TO 17
C   RETURN TO START POINT
      GO TO 11
C   MAKE PREV CLASS '1' STORE INDEX AND QUIT
13    K3=I
      IF(I.EQ.0)K3=1
      IF(WORD(K1-1,4).EQ.TWO)GO TO 109
110   K4=K1-1
      WORD(K1-1,4)=ONE
      GO TO 16
C   IF CLASS '2' FIND IF IT IS ENDING WITH 'ING'
109   K1=K1-1
      CALL STRIP(K1,3)
      IF(ENDING.EQ.ING)K1=K1+1
      GO TO 110
```

```
C    STORE POINTERS
14     K3=I
       IF(I.EQ.0)K3=1
       K4=K1
       GO TO 16
C    IF INCOMING WORD IS '2' BUT NOT MODIFEE VERB QUIT
15     CALL STRIP(K1,3)
       IF(ENDING.EQ.ING)GO TO 12
       CALL STRIP(K1,2)
       IF(ENDING.EQ.ED)GO TO 12
       GO TO 13
C    IF SOME IN BETWEEN WORDS ARE BLANK MAKE THEM CLASS '3'
16     DO 100 J=K3,K4
       IF(WORD(J,4).EQ.BLANK)WORD(J,4)=THREE
100    CONTINUE
       RETURN
17     WORD(K1,4)=ONE
       GO TO 14
C    GROUP 'B' ANALYSIS BEGINS HERE
201    K1=I
21     K1=K1+1
C    IF IT IS GROUP 'B' GO TO NEXT WORD
C    IF IT IS CLASS '2' QUIT
       IF(WORD(K1,4).EQ.BLANK)GO TO 22
       IF(WORD(K1,4).EQ.B)GO TO 21
       IF(WORD(K1,4).EQ.TWO)RETURN
C    IS IT SINGLE GROUP OR MUL GRAMMER CLASS
       K2=INTGER(AND(WORD(K1,4),MSK1))
       IF(K2.EQ.MSK4)GO TO 24
C    FIND IF ENDING IS 'ING' OR 'ED'
       CALL STRIP(K1,3)
       IF(ENDING.EQ.ING)GO TO 22
       CALL STRIP(K1,2)
       IF(ENDING.EQ.ED)GO TO 22
C    DOES WORD BELONG TO SPECIAL GROUP
       DO 200 J=1,4
       IF(WORD(K1,4).EQ.TAB(J))GO TO 23
200    CONTINUE
C    SEPERATE THE TO GRAMMATICAL UNITS
       WRITE(99,1000)WORD(K1,4)
       READ(99,1001)K3,K4
C    IS ANY BELONGING TO GROUP CLASS '2'
       IF(K3.EQ.B.AND.K4.EQ.F)GO TO 25
       IF(K3.EQ.TWO.OR.K4.EQ.TWO)GO TO 22
       RETURN
24     WORD(K1-1,4)=TWO
       RETURN
25     IF(WORD(K1+1,4).EQ.BLANK)GO TO 23
       WORD(K1,4)=F
       GO TO 24
```

```
22    WORD(K1,4)=TWO
      RETURN
23    WORD(K1,4)=B
      GO TO 21
C     GROUP 'C' ANALYSIS BEGINS AT THIS POINT
301   K1=I
      IF(WORD(K1+1,4).EQ.BLANK)GO TO 33
C     SINGLE WORD GROUP CHECK PREV IS 'B' OR '2' ENTER '2' OR '3' RESPLY
      K2=INTGER(AND(WORD(K1+1,4),MSK1))
      IF(K2.EQ.MSK4)GO TO 33
      WRITE(99,1000)WORD(K1+1,4)
      READ(99,1001)K3,K4
C     IF IT IS '3' OR'2' SET   NEXT GROUP ACCORDINGLY
      IF(K3.EQ.THREE.OR.K4.EQ.THREE)GO TO 33
      IF(K3.EQ.TWO.OR.K4.EQ.TWO)GO TO 33
      RETURN
33    IF(WORD(K1-1,4).EQ.B)GO TO 31
      IF(WORD(K1-1,4).EQ.TWO)GO TO 32
      RETURN
31    WORD(K1+1,4)=TWO
      RETURN
32    WORD(K1+1,4)=THREE
      RETURN
C     GROUP 'D' ANALYSIS
401   K1=I-1
41    K1=K1+1
C     NEXT WORD ALSO GROUP 'D' GO BACK TO INCREMENT
C     NEXT THREE OR FOUR GO TO QUIT
      IF(WORD(K1,4).EQ.D)GO TO 41
      IF(WORD(K1+1,4).EQ.BLANK)GO TO 44
      IF(WORD(K1+1,4).EQ.THREE.OR.WORD(K1+1,4).EQ.FOUR)RETURN
C     GET THE TWO GRAMMATICAL ENTRIES
      WRITE(99,1000)WORD(K1+1,4)
      READ(99,1001)K3,K4
      IF(K3.EQ.D.OR.K4.EQ.D)GO TO 42
      IF(K3.EQ.THREE.AND.K4.EQ.FOUR)RETURN
      IF(K4.EQ.THREE.AND.K3.EQ.FOUR)RETURN
C     CORRESPONDING TO '3' AND '4' MAKE ENTRIES
      IF(K3.EQ.THREE.OR.K4.EQ.THREE)GO TO 43
      IF(K3.EQ.FOUR.OR.K4.EQ.FOUR)GO TO 444
      RETURN
42    WORD(K1+1,4)=D
      RETURN
444   WORD(K1+1,4)=FOUR
      RETURN
43    WORD(K1+1,4)=THREE
      RETURN
44    WORD(K1+1,4)=THREE4
      RETURN
C     CONJUNCTION CLASS GROUP 'E' ANALYSIS
```

```
C     STORE NEXT AND PREV IN TEMP AREA
501    K1=I
51     K2=WORD(K1-1,4)
       K3=WORD(K1+1,4)
C     FIND INDIVIDUAL DOUBLE GRAMMER GROUPS
       WRITE(99,1000)K2
       READ(99,1001)K5,K6
       WRITE(99,1001)K3
       READ(99,1001)K7,K8
C     GRAMMER OF NEXT AND PREV SAME QUIT
       IF(K2.EQ.K3)RETURN
C      IF THE PREV IS UNIQUE GO TO 52
       K4=INTGER(AND(MSK2,K2))
       IF(K4.EQ.K2)GO TO 52
C     IF NEXT IS UNIQUE GRAMMER GO TO 53
       K4=INTGER(AND(MSK2,K3))
       IF(K4.EQ.K3)GO TO 53
       RETURN
C     IF PREV IS CLASS '1'GO TO 54
C     MATCH PART OF NEXT WITH PREV ENTER
C     IF BLANK MAKE NEXT SAME AS PREV
52     IF(K2.EQ.ONE)GO TO 54
       IF(K2.EQ.K7.OR.K2.EQ.K8)WORD(K1+1,4)=K2
       IF(K3.EQ.BLANK)WORD(K1+1,4)=K2
       RETURN
C     IF NEXT 'A' OR '1' THEN PREV IS '1'
C     MATCH PREV SUBSET WITH NEXT ENTER NEXT
C     IF PREV BLANK ENTER PREV SAME AS NEXT
53     IF(K3.EQ.ONE.OR.K3.EQ.A)GO TO 55
       IF(K3.EQ.K5.OR.K3.EQ.K6)WORD(K1-1,4)=K3
       IF(K2.EQ.BLANK)WORD(K1-1,4)=K3
       RETURN
C     SETTING OF GROUP '1'
54     IF(K3.EQ.ONE2)GO TO 58
59     IF(K7.EQ.TWO)GO TO 56
       IF(K8.EQ.TWO)GO TO 57
       IF(K3.EQ.BLANK)WORD(K1+1,4)=ONE3
       RETURN
58     CALL STRIP(K1+1,3)
       IF(ENDING.EQ.ING)GO TO 59
       WORD(K1+1,4)=ONE
       RETURN
56     WORD(K1+1,4)=K8
       RETURN
57     WORD(K1+1,4)=K7
       RETURN
55     WORD(K1-1,4)=ONE
       RETURN
C      GROUP F OR PREPOSITION ANLYSIS
601    K1=I
```

```
      IF(WORD(K1+1,4).EQ.BLANK)GO TO 11
      IF(WORD(K1+1,4).EQ.A.OR.WORD(K1+1,4).EQ.ONE)RETURN
C     FETCH THE DOUBLE GROUPS
      WRITE(99,1000)WORD(K1+1,4)
      READ(99,1001)K3,K4
C     IF IT IS 'A1' OR '1A' GROUP   OR 'A'  SET AND RETURN
      IF(K3.EQ.ONE.AND.K4.EQ.A)GO TO 61
      IF(K3.EQ.A.AND.K4.EQ.ONE)GO TO 62
      IF(A.EQ.K3.OR.K4.EQ.A)GO TO 62
C     GOTO ANALYSIS OF GROUP A
      GO TO 11
62    WORD(K1+1,4)=A
      RETURN
61    WORD(K1+1,4)=ONE
      RETURN
C     GROUP ANALYSIS FOR G(PARTICULAR WORD FRAME OF 'DO')
701   K1=I
      WRITE(99,1000)WORD(K1+1,4)
      READ(99,1001)K2,K3
      IF(WORD(K1+1,4).EQ.BLANK)GO TO 73
      IF(WORD(K1+1,4).EQ.ONE.OR.WORD(K1+1,4).EQ.A)RETURN
      IF(WORD(K1+1,4).NE.ANOT)GO TO 71
      WORD(K1+1,4)=C
      RETURN
71     IF(WORD(K1+1,4).NE.MSK3)GO TO 72
      IF(K2.EQ.A.OR.K3.EQ.A)WORD(K1+1,4)=A
      RETURN
72    IF(K2.EQ.ONE.OR.K3.EQ.ONE)WORD(K1+1,4)=ONE
      RETURN
73    WORD(K1+1,4)=ONE
      RETURN
C     GROUP H ANALYSIS (WORD 'THERE' BREGINNING THE SENTENCE
801   IF(I.EQ.1)GO TO 81
      IF(WORD(I-1,4).EQ.TWO)RETURN
      IF(WORD(I-1,4).EQ.BLANK)WORD(I-1,4)=TWO
      WRITE(99,1000)WORD(I-1,4)
      READ(99,1001)K2,K3
      IF(K2.EQ.TWO.OR.K3.EQ.TWO)WORD(I-1,4)=TWO
      RETURN
81    IF(WORD(I+1,4).EQ.TWO)RETURN
      IF(WORD(I+1,4).EQ.BLANK)WORD(I+1,4)=TWO
      WRITE(99,1000)WORD(I+1,4)
      READ(99,1001)K2,K3
      IF(K2.EQ.TWO.OR.K3.EQ.TWO)WORD(I+1,4)=TWO
      RETURN
C     GROUP I ANALYSIS (THESE ARE QUESTION WORDS IN THE BEGINNING)
901   IF(I.NE.1)RETURN
      IF(WORD(I+1,4).EQ.G.OR.WORD(I+1,4).EQ.TWO)RETURN
      IF(WORD(I+1,4).EQ.BLANK)WORD(I+1,4)=TWO
      WRITE(99,1000)WORD(I+1,4)

      READ(99,1001)K2,K3
      IF(K2.EQ.TWO.OR.K3.EQ.TWO)WORD(I+1,4)=TWO
      RETURN
      END
```

ẸE-1972-M-GAN-PRO